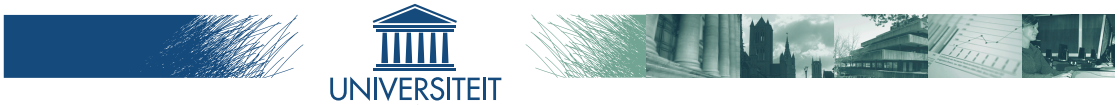


Heuristic algorithms for payment models in project scheduling

Pieter Leyman

Dissertation submitted to the Faculty of Economics and Business Administration,
Universiteit Gent, in fulfillment of the requirements for the degree of
Doctor of Applied Economic Sciences

"All truths are easy to understand once they are discovered; the point is to discover them."

Galileo Galilei

FACULTY OF **ECONOMICS**
AND BUSINESS ADMINISTRATION

Promotor:

Prof. dr. Mario Vanhoucke

Doctoral jury:

Prof. dr. Marc De Clercq	Universiteit Gent, Dean
Prof. dr. Patrick Van Kenhove	Universiteit Gent, Academic Secretary
Prof. dr. Tarik Aouam	Universiteit Gent
Prof. dr. José Coelho	Universidade Aberta, Lisbon (Portugal)
	INESC, Porto (Portugal)
Prof. dr. Broos Maenhout	Universiteit Gent
Prof. dr. Geert Poels	Universiteit Gent
Prof. dr. Vincent Van Peteghem	EDHEC Business School, Lille (France)

Dankwoord

Mijn doctoraat is tot stand gekomen dankzij de bijdragen en steun van een aantal bijzondere mensen, want het schrijven van dit boek is absoluut geen solotraject geweest. Elk op hun eigen manier hebben zij er toe bijgedragen dat ik volhardde en moed bleef houden, want telkens opnieuw beginnen na een tegenslag is het moeilijkste dat er bestaat. Ik heb gepoogd om dan ook iedereen die me de afgelopen 4 jaar bijgestaan heeft op te nemen in dit dankwoord. Om enigszins te vermijden dat ik iemand vergeet wil ik, net als een sympathieke voorganger van mij, reeds iedereen die dit leest bedanken. Ongetwijfeld heb jij bijgedragen aan mijn doctoraat of mijn leven daarbuiten de afgelopen jaren, ofwel omdat je interesse hebt in het gevoerde onderzoek of in mij als persoon. Daarnaast wil ik me ook nog tot enkele personen in het bijzonder richten, die me, ondanks mijn koppige, soms brompotachtige en niet altijd even communicatieve karakter, telkens opnieuw gesteund hebben.

In academische omgeving, dien ik in de eerste plaats mijn promotor Mario te bedanken. Zonder zijn enthousiasme en niet–aflatende stroom aan suggesties en opmerkingen zou ik nooit geraakt zijn waar ik nu sta. Onze discussies hebben me in staat gesteld mijn inzichten te verfijnen en te streven naar telkens beter en meer. Verder gaat mijn dank uit naar alle leden van mijn examencommissie, Marc, Patrick, Tarik, José, Broos, Geert en Vincent. Dankzij hun interesse en waardevolle commentaren ben ik er in geslaagd om dit doctoraat succesvol af te werken. I would like to specifically thank the members of my reading committee (Tarik, José, Broos, Geert and Vincent), for the many questions and insightful suggestions to further improve the quality of my dissertation. Daarnaast dien ik ook het "Bijzonder Onderzoeksfonds"(BOF) en de Nationale Bank van België te bedanken voor de financiële ondersteuning die dit doctoraat mogelijk gemaakt heeft (contractnummer BOF12GOA021). De ondersteuning van Martine en Machteld liet me dan weer toe om me voornamelijk te focussen op mijn onderzoek en andere taken, terwijl zij de bijhorende administratie voor hun rekening namen. Ook met vragen rond lessenroosters en examens kon ik keer op keer bij hen terecht.

Het leven op kantoor zou een stuk saaier (en ook minder productief) geweest zijn zonder

toffe collega's. Ik wens Christophe, Jeroen C., Len en Mathieu te bedanken om me goed op te vangen toen ik startte, en een breed scala aan vragen telkens opnieuw met hetzelfde enthousiasme te beantwoorden. Daarnaast wil ik ook Eva en Thomas bedanken, omdat hun regelmatige aanwezigheid de lunchpauzes (en het eten van De Brug) een stuk aangenamer maakten. Ook de gebeurlijke koffie of thee in de OR hielp vaak als nodige opkikker. Specifiek, wil ik Christophe nog bedanken omdat hij de rol als mijn peter op zich nam, met vaak nuttige maar soms ook grappige suggesties en tips tot gevolg. Mijn dank gaat tevens uit naar de jongere collega's Jonas, Laura, Annelies, Jeroen B., Tom en Mick, die de afgelopen twee jaar de trend van gezamenlijke middag- en namiddagpauzes hebben helpen in stand houden. Ook werd dankzij hen het concept van vrijdag na het werk pinten te gaan drinken op gang getrokken en in stand gehouden, waarbij ook Jordy en Louis–Philippe regelmatig present tekenden. Bovendien wil ik Jordy en Lynn bedanken om me beter wegwijs te maken in Barcelona, onder meer door enkele uitstekende suggesties te doen met betrekking tot bezienswaardigheden en eetgelegenheden. Een speciale bedanking gaat uit naar Annelies en Jeroen B. voor het vele werk dat zij verricht hebben aan het groepswerk voor het vak Applied Operations Research, en de vlotte en constructieve samenwerking doorheen het hele traject. Jeroen B., Jonas en Len wil ik nog eens extra bedanken omdat ik bij hen als medeplanners altijd terecht kon met vragen groot en klein, wat dikwijls leidde tot geanimeerde maar altijd productieve discussies. Dank aan Niels, omdat zijn inzichten en onze discussies me hebben in staat gesteld om een volwaardig hoofdstuk van dit boek te schrijven, vertrekkend van zijn masterproef. Keer op keer stelde hij enthousiast zijn werk voor en bracht hij waardevolle inzichten aan.

Ik wil mijn vrienden Jonas, Mathias, Michaël en Steven bedanken omdat ze me steeds weer hielpen om te ontspannen. Of het nu was door een vaak voorkomend avondje film (gaande van de nieuwste Marvel prent tot Star Wars), samen iets gaan drinken of eten, of een VOSEKO event, ze waren telkens paraat. Ze lieten me toe om mijn hart te luchten indien nodig, of zorgden voor geanimeerde discussies over allerhande absurde en minder absurde onderwerpen, wat voor de nodige ontspanning zorgde. Bovendien waren ze ook keer op keer benieuwd naar de voortgang van mijn doctoraat. Jonas, je was de afgelopen jaren naast een collega tevens een goede vriend, die altijd klaar stond met goed advies, zowel in de werksfeer als privé. Je luisterend oor en goede smaak voor bier vormden de basis van menig goed gesprek. Michaël, als medegamer en filmnerd kon ik bij jou in het bijzonder terecht met beslommeringen rond het einde van Bioshock Infinite, en de vraag of Batfleck nu een goed idee was of niet. Je hebt me ook geïntroduceerd in de fantastische werelden (of moet ik zeggen universa?) van Mass Effect en Dragon Age, waarvoor ik je enorm dankbaar ben. Daarnaast wekte je enthousiasme voor alles (werk-, gaming-, filmgerelateerd en nog veel meer) vaak aanstekend, en was je altijd beschikbaar voor een

goede babbel. Mathias, jouw neiging om alles ludiek en genuanceerd te bekijken zorgde vaak voor de nodige humor, zeker als de discussies op café iets te serieus werden. Je hebt me geleerd dat het zowel professioneel als privé kan helpen om regelmatig een stap terug te zetten, me meer te ontspannen, en de dingen te durven nemen zoals ze komen. Steven, jouw gevoel voor humor lag de afgelopen jaren vaak aan de basis van menig amusant gesprek. Je oprechte interesse in en enthousiasme voor politiek (en een bepaalde partij) kwamen ook regelmatig aan bod, waardoor je me hebt bijgebracht dat het belangrijk is om een zekere vorm van idealisme te behouden.

Tot slot, wil ik me tot mijn familie richten. Om te beginnen, wil ik mijn grootouders bedanken, niet alleen voor de afgelopen 4 jaar, maar ook voor de 24 jaar daarvoor. Het doet me enorm veel pijn dat jullie niet allemaal de voltooiing van mijn doctoraat kunnen meemaken, maar weet dat het me zonder jullie steun niet zou gelukt zijn. Daarnaast gaat mijn dank uit naar nonkel Edwin, die steeds zijn interesse liet blijken in mijn onderzoek, en me ook aanmoedigde telkens als ik het wat moeilijker had. Vervolgens, wil ik iedereen thuis bedanken voor jullie steun, die vaak tot op mijn werkplek voelbaar was, en in het bijzonder mijn broer Gert en mijn ouders. Gert, je hebt me de afgelopen jaren telkens onvoorwaardelijk gesteund en je was ook steeds beschikbaar als ik een opbeurende babbel nodig had. Onze, vaak ludieke, discussies gaande van werkomstandigheden en "rare"collega's tot bij welke voetbalclub een zekere (oude) rot het veld nu weer onveilig maakte, waren een aangename afleiding. Je interesse in actiefilms en -reeksen zoals Jason Bourne en The Blacklist, samen met mijn hernieuwde bekommernis om het wel en wee van de Rode Duivels, zorgde dan weer voor het amusante avondje uit of voor de tv. Daarnaast was je ook steeds bereid om mijn teksten (inclusief dit dankwoord!) na te lezen op mogelijke taal- en tikfouten. Mama, Papa, er bestaan geen woorden die mij in staat stellen om uit te drukken wat jullie voor me betekenen en betekend hebben de afgelopen jaren. Desalniettemin, ga ik toch een poging ondernemen. Zonder jullie onvoorwaardelijke steun, vertrouwen en liefde zou ik dit doctoraat nooit hebben kunnen voltooien, en zou ik niet de persoon zijn die ik vandaag ben. Jullie zijn er altijd voor mij geweest, bij ups en al helemaal bij downs. Als het wat minder ging, slaagden jullie er keer op keer in om me te motiveren en te zorgen dat ik volhardde. Jullie zijn mijn rots in de branding, mijn toeverlaat, mijn thuis, en dat zullen jullie ook altijd blijven! Vanuit de grond van mijn hart: bedankt!

Pieter Leyman
Gent, 2 september 2016

Nederlandstalige samenvatting

Veronderstel dat het stadsbestuur van Gent beslist om een nieuwe brug te laten bouwen over de Leie. Het doel is om de verkeersdrukte in het centrum te verminderen, en als gevolg stelt het stadsbestuur een uiterste datum voorop voor de ingebruikname van de brug. Op basis van de specificaties gaat een aannemer vervolgens het benodigd aantal midellen (bv. machines, mankrachten) bepalen en een project schema opstellen. Dit schema bevat de start- en eindtijden van elke activiteit (bv. het gieten van beton voor de funderingen), en houdt rekening met de opgelegde middelen beperkingen en de volgorde waarin de activiteiten uitgevoerd moeten worden (bv. de oevers moeten afgegraven worden alvorens de funderingen kunnen gegoten worden). Terwijl de doelstelling van de klant (het stadsbestuur) duidelijk is, ze willen namelijk dat de brug op tijd voltooid is, is de doelstelling voor de aannemer een stuk minder voor de hand liggend. Zou de aannemer best de totale duurtijd minimaliseren, de totale kost minimaliseren, of de netto actuele waarde (NAW) maximaliseren, etc.?

Veronderstel dat de aannemer twee schema's kan opstellen. Het eerste schema minimaliseert de project duurtijd, heeft een totale duur die zorgt voor een voltooiing van de brug 6 weken voor de uiterste datum, en heeft een NAW van €1 miljoen. Het tweede schema, daarentegen, maximaliseert de NAW, met als gevolg een voltooiing van de brug die samenvalt met de uiterste datum en een NAW van €1,2 miljoen. Het tweede schema kan bekomen worden door het later inplannen van sommige activiteiten, gegeven de opgelegde beperkingen, vertrekkend van het eerste schema. Als we er van uitgaan dat er voldoende marges zijn voorzien tegen mogelijke vertragingen, dan zou de aannemer logischerwijs het tweede schema verkiezen, omdat dit financieel aantrekkelijker is. De cruciale vraag wordt dan hoe het tweede schema op een efficiënte en effectieve manier kan bekomen worden, vertrekkend van het eerste schema. Deze doctoraatsthesis heeft als doel om algoritmen te ontwikkelen die de NAW van projecten optimaliseren onder verschillende omstandigheden.

In hoofdstuk 1 geven we een beknopte inleiding tot project management, alvorens project planning als deelgebied toe te lichten. We bespreken kasstromen in project planning, op basis van betaalmomenten en -groottes, en geven een overzicht van de volgende

hoofdstukken, vertrekkend van vijf onderzoeksvragen.

Hoofdstuk 2 gaat dieper in op NAW optimalisatie gegeven volgorderelatie- en middelenbeperkingen. Daarnaast, veronderstellen we dat zowel de kasinstromen (ontvangen van de klant) als de kasuitstromen (betaald aan onderaannemers) gebeuren op het einde van elke activiteit. Op deze manier, is de grootte van de betalingen vastgelegd door de klant en komt deze overeen met de kasstromen per activiteit. De betaalmomenten, daarentegen, hangen af van de concreet gekozen eindtijden per activiteit, gekozen door de aannemer in het gebruikte schema.

Hoofdstukken 3 & 4 behandelen andere betalingsmodellen, waar de klant de betaalmomenten vastlegt in plaats van de groottes van de betalingen. De klant kan bijvoorbeeld vooropstellen dat de aannemer elke maand betaald wordt, terwijl de grootte van de betalingen afhangt van het werk dat uitgevoerd is door de aannemer in elke maand. Ook kan de klant opleggen dat de betalingen afhangen van de voortgang van het project in functie van gecreëerde waarde of van de gemaakte kosten door de aannemer. In beide laatste gevallen kan de aannemer dan zowel de betaalmomenten als -hoeveelheden bepalen, maar zijn er andere beperkingen opgelegd. In zowel hoofdstuk 3 als 4, gaan we bovendien verschillende alternatieve uitvoeringsmogelijkheden voor elke activiteit in beschouwing nemen. Deze alternatieven zorgen voor verschillende combinaties van duurtijd en middelenvraag per activiteit, uit de welke telkens één dient gekozen te worden per activiteit. Als gevolg hiervan heeft de aannemer een grotere flexibiliteit bij het plannen.

In hoofdstuk 5 introduceren we kapitaal als middel voor de aannemer. De bijkomende beperking met betrekking tot kapitaal stelt dat dit nooit negatief mag worden tijdens het project. Anders heeft de aannemer immers geen geld meer om het project verder uit te voeren. Het beschikbare kapitaal hangt af van de initieel beschikbare hoeveelheid, maar ook van de kas in- en uitstromen, die respectievelijk zorgen voor toenames en afnames. We gebruiken een algemeen model om de beschikbaarheid van kapitaal in een project te bepalen en beïnvloeden. Hierbij is het cruciaal om de activiteiten zodanig te plannen dat de volgorde toelaat dat er telkens voldoende kapitaal beschikbaar is om met latere activiteiten te starten. Belangrijke inzichten worden aangeboden voor de aannemer, opdat deze kan bepalen wat het belangrijkst is bij het plannen van het project, namelijk de NAW optimaliseren of een gunstige kaspositie waarborgen.

Hoofdstuk 6 zorgt voor de integratie van de middelenbeschikbaarheid in het planningsproces. Als gevolg, kan de NAW van het project geoptimaliseerd worden samen met de gebruikskosten van de middelen, eerder dan beide los van mekaar te behandelen. Belangrijk hierbij is dat door de integratie van beide stappen, dit een efficiënter middelengebruik bij de aannemer toelaat. Deze gaat immers enkel het benodigd aantal middelen aan een project toekennen, gegeven de opgelegde beperkingen en de NAW doelstelling, eerder dan

een (ruwe) schatting maken zoals in eerdere hoofdstukken het geval was.

Tot slot, geven we in hoofdstuk 7 een samenvatting van het onderzoek, samen met de belangrijkste bevindingen, en formuleren we een antwoord op de gestelde onderzoeksvragen. We geven ook aan wat mogelijke pistes zijn voor toekomstig onderzoek rond NAW optimalisatie binnen project planning.

Table of Contents

List of Figures

List of Tables

1

Introduction

1.1 What does project management entail?

A broad definition of project management can be given as follows: "Project management is the discipline of planning, organizing and managing resources to bring about the successful completion of specific project goals and objectives" (Vanhoucke, 2012). From this definition of project management, three crucial characteristics can be determined:

- **A project has specific goals and objectives.** With each project we want to achieve certain objectives, e.g. build a new bridge crossing the Leie for both motorized vehicles and pedestrians.
- **A project is finite in runtime.** A project has a clear start and end, which is in stark contrast with e.g. a production line. E.g. the work on a new bridge has a predefined number of steps which have to be completed. Once all steps involved are finished, the project is terminated.
- **Project management involves planning of resources.** If we aim to build the aforementioned bridge, we have to manage resources such as workers and machinery. We have to determine when each resource is required, at what level and for how long.

As a general framework for project management, we briefly discuss two possibilities. Together, both options allow for broad overview of the steps involved in project management as a whole. The first framework is Dynamic Scheduling, of which the three dimensions are displayed in figure 1.1 (Vanhoucke, 2012), and summarized along the following lines.

- **Baseline Scheduling:** establish a start and end time for each of the activities in the project. Relations between the activities and the resource demands of each activity are taken into account, given a predefined scheduling objective.
- **Schedule Risk Analysis:** consider the strong and weak points of the baseline schedule, and determine the impact of unforeseen events on the project schedule and objective.
- **Project Control:** determine the project performance during its execution both in terms of duration and cost. Update and adapt the project schedule based on information from the Baseline Scheduling and Schedule Risk Analysis parts.

The second framework is the Project Life Cycle (PMBOK, 2004), as shown in figure 1.2. Six distinct phases can be observed.

- **Concept:** determine the need for and general objective of a project.

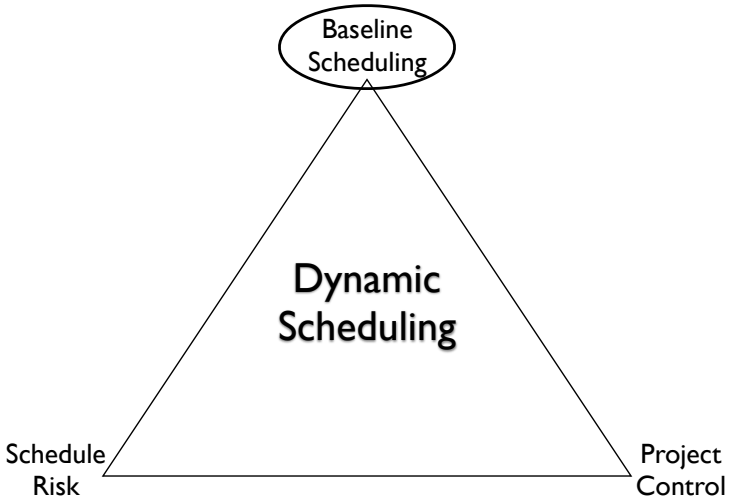


Figure 1.1: Three dimensions of Dynamic Scheduling (Vanhoucke, 2012).

- **Definition:** delineate the individual project activities, the relations between the activities, and the specific goals of the project.
- **Scheduling:** construct a timetable for the project based on the requirements of the definition phase.
- **Execution:** implementation of the project activities.
- **Control:** evaluate the performance and take corrective actions if needed (feedback loop in figure 1.2).
- **Termination:** complete the project and do a final evaluation of its performance.

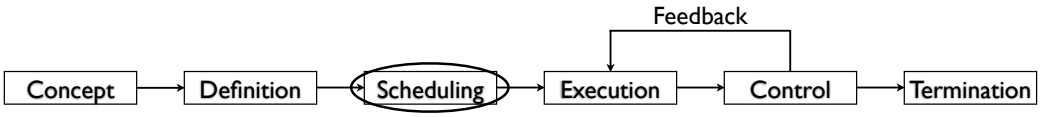


Figure 1.2: Project Life Cycle (PMBOK, 2004).

1.2 Research contribution

In this section, we first provide details with respect to some general concepts on project scheduling and cash flows. The research contribution of this PhD is subsequently high-

lighted, based on the models used for payments. Finally, an overview of the different chapters is included.

1.2.1 General concepts in project scheduling

The focus of this PhD is on project scheduling, which implies that our research revolves around the Baseline Scheduling part of Dynamic Scheduling (figure 1.1) and around the Scheduling phase of the Project Life Cycle (figure 1.2). A project can typically be represented by a directed graph or network $G(N, A)$ with N used for the project activities or nodes and A the precedence relations or arcs between the nodes N. We employ the activity–on–the–node (AoN) representation and assume a time–lag of zero for the precedence relations. No preemption of activities is allowed. Each activity i ($i \in N = \{1, \ldots, n\}$) has a duration d_i, and a resource demand of r_{ik} of type k. Each renewable resource type k ($k \in R = \{1, \ldots, |R|\}$) has a limited availability of a_k. Additionally, a start dummy 0 and end dummy $n + 1$ are included. A schedule corresponds with an assignment of finish times for all activities.

The resource–constrained project scheduling problem (RCPSP) aims to minimize project makespan subject to precedence and renewable resource restrictions. The RCPSP has been extensively discussed in literature, but it is a relatively basic problem with assumptions which may not always be applicable in practice (Hartmann and Briskorn, 2010). However, several extensions have been proposed and discussed in literature. Hartmann and Briskorn (2010) provide a recent overview on extensions of the RCPSP, and distinguish between activity concepts, temporal constraints, resource constraints and the objective.

- **Activity concepts:** No preemption, or interruption of an activity once it has been started, is allowed in the RCPSP. Other activity concepts aside from preemption are setup costs and multiple activity modes. In this PhD, we assume no preemption is allowed, but include multiple activity modes in several of the chapters.

- **Temporal constraints:** The RCPSP assumes that only minimal time lags of zero between activities are imposed, i.e. an activity can start once all of its predecessors have been completed. This is also called a finish–start precedence relation with a minimal lag of zero. Several alternatives exist in literature, e.g. maximum time lags and release dates, but in the research presented here, we always assume minimal time lags of zero.

- **Resource constraints:** In the RCPSP only one type of resources is used, namely renewable ones. These resources are called renewable because their availability equals full capacity in every time period. Examples are manpower and machine hours. We

include renewable resources, but also discuss two other types of resources. Non–renewable resources have a limited fixed available for the entire project, and only need to be considered in a multi–mode context. Examples are the available hours of a manager for a project and raw materials available. Cumulative resources have a variable availability, which is not necessarily the same after the completion of an activity (Neumann and Schwindt, 2002). Examples of a cumulative resource are inventory and capital.

- **Objective:** As stated earlier, the objective of the RCPSP is makespan minimization. However, several other objectives exist such as total cost minimization, resource idle time minimization and resource levelling. In this PhD, the focus lies on net present value (NPV) optimization, subject to different types of restrictions. As a result, a cash inflow $c_{i,in}$ (> 0) and a cash outflow $c_{i,out}$ (< 0) are assigned to each activity.

1.2.2 Cash flows in project scheduling

In NPV optimization, the activity cash flows are discounted based on a discount rate and the occurrence of the cash flows. The timing and size of these cash flows, however, often depend on a negotiation between the client and contractor of a project. The client is the party receiving the benefits and paying for the execution of the project, whereas the contractor is the party responsible for the execution of the project. In this PhD, we assume that the negotiations between both parties have been completed, and we employ the contractor's point of view. Hence, the objective is to optimize the contractor's NPV, given the project characteristics (i.e. activity, temporal and resource specifications) and restrictions with respect to the timing and/or size of cash flows. Additionally, a project deadline is imposed, since otherwise negative cash flows may be delayed indefinitely, resulting in the project never being completed.

The research presented here, tackles different models for the timing and size of both cash in– and outflows. The *timing* of cash flows determines when payments are received from the client (cash inflows) and when payments are due to e.g. subcontractors (cash outflows), whereas the *size* regulates the amount to be received or paid respectively. The distinction can be made between three general classes of payment models:

1. The size of payments is determined in advance, but the timing depends on the schedule constructed. In this case, the contractor can influence the occurrence times of payments by changing the finish times of activities, but the size of the payments always equals the activity cash flows. This model is called payments at activities' completion times (PAC) in literature.

2. The timing of payments is set, but the size depends on the schedule, namely on the work done since the previous payment. This model assumes payments occur at regular intervals, e.g. every 2 weeks, and is called progress payments (PP). A variant exists in which payments occur at irregular intervals, which is called payments at event occurrences (PEO). In both cases, payments are linked to the progress of the project in terms of time. An extreme case is the lump sum payment (LSP) model, in which only one payment occurs upon project completion.

3. Both the size and timing depend on the schedule of the contractor. In this case, several possibilities exist. First, the contractor is free to optimize their NPV as long as a predefined number of payments is guaranteed (see e.g. Dayanand and Padman (1997)). Second, the contractor may impose that progress is measured differently than in the PP and PEO models, namely based on the created value for the client or based on the cost incurred by the contractor (see e.g. He et al. (2009b)). These models are called the progress–based payment pattern (PBPP) and expense–based payment pattern (EBPP) respectively.

Figure 1.3 provides an overview of the models discussed in this book, for both cash in– and outflows, and is repeated at the start of each chapter. The figure shows the distinction between the three general classes, namely timing, size and timing & size of cash flows. We apply five different models for the cash inflows of activities. For cash outflows, however, in literature only the PAC model is used. As a result, we start with this model for cash outflows, but introduce a general model for cash outflows as part of capital management for the contractor in chapter 5, whereas resource usage costs are included in chapter 6. The LSP model is not discussed in this PhD, because this model corresponds with a RCPSP in which all activities with a negative cash flow should be scheduled as late as possible, given the obtained makespan. Hence, we believe no additional research is needed for the LSP model.

1.2.3 Research questions

The complex payment models highlight the need for specialized scheduling techniques, which are currently lacking in literature, in particular for problems of a realistic size (e.g. 50 or more activities). Hence, the goal of this PhD is as follows: we aim to analyze the models discussed above in detail, propose new scheduling techniques to handle the inherent complexity, and make the models more realistic by including e.g. capital management on the side of the contractor. We focus on heuristic optimization to allow for a greater degree of complexity and to be able to construct good schedules in reasonable time. The following research questions (RQ) summarize the goal of the PhD:

	Timing	**Size**	**Timing & size**
Cash in	Payments at activities' completion times (PAC)	Progress payments (PP)	Progress based payment pattern (PBPP)
		Payments at event occurences (PEO)	Expense based payment pattern (PBPP)
Cash out	Payments at activities' completion times (PAC)	Resource usage costs	General capital model

Figure 1.3: Overview of the research on project scheduling with NPV optimization.

- **RQ1:** What is a good scheduling technique to use for the timing of cash flows?
- **RQ2:** What is a good scheduling technique to use for the size of cash inflows?
- **RQ3:** How can these schedulers be applied in case of different types of activity trade–offs?
- **RQ4:** How can cash outflows and capital be managed for the contractor, under different assumptions?
- **RQ5:** How can resource usage costs be optimized and integrated in NPV optimization?

With each of the proposed methodologies in the different chapters we aim to answer one or more of these research questions. The added value from a methodological point of view lies in on one or more of the following: solution representation (R), scheduling techniques (S), and metaheuristics (M). The contents of the different chapters is summarized in table 1.1, and a more detailed chapter overview is given in section 1.2.4.

Chapter	**Focus**	**Methodology**		
		R	S	M
2	RQ1		X	X
3	RQ2 & 3		X	
4	RQ2 & 3	X	X	
5	RQ4		X	X
6	RQ5	X	X	

Table 1.1: Contents of different chapters.

1.2.4 Chapter overview

In **chapter 2** (Leyman and Vanhoucke, 2015), we extend the RCPSP by including the PAC model for both the cash in– and outflows, and aim to answers **RQ1**. As a result, we optimize the project NPV subject to precedence and resource restrictions, and a deadline. We propose a new **scheduling technique**, which moves sets of activities based on their combined or cumulative NPV. These sets can be constructed based on the project network or based on the schedule under consideration. This distinction allows the procedure to focus on either the precedence or resource restrictions. Two variants of a **metaheuristic** are analyzed and tested. We successfully show the added value of our scheduler by outperforming two benchmarks from literature. Both the methods employed and the results obtained in this chapter serve as a starting point for the subsequent chapters.

In **chapter 3** (Leyman and Vanhoucke, 2016b), we build upon the previous chapter and focus on **RQ2** and **RQ3**. The PP and PEO models for the cash inflows are included on top of the PAC model, whereas cash outflows are paid at activity completion in all three cases. A more complex variant of the **scheduler** of chapter 2 is introduced, which considers the peaks in the NPV profiles of activities. The problem is also extended to its multi–mode variant, by including trade–offs between different execution modes for each activity. Each activity mode considers different requirements for both renewable and non–renewable resources and provides a different activity duration. The inclusion of non–renewable resources because of the different activity modes, increases complexity with respect to the project resources. The results show the strong added value of the scheduling techniques, both in a single–mode and multi–mode context.

In **chapter 4** (Leyman et al., 2016), we discuss three variants to the PAC model of chapter 2, namely the PP, PBPP and EBPP models. Unlike in the previous chapters, the payment models are not applied in the context of the RCPSP, but rather in a discrete time/cost trade–off problem (DTCTP). The activity modes in the DTCTP consider only the activity duration and a single non–renewable resource, and can as such be seen as a more specific application of multi–mode scheduling. Given the nature of the three models and the existence of different activity modes in the DTCTP, we again aim to answer **RQ2** and **RQ3**, but consider these questions from a different angle than in chapter 3. Aside from the models, the main focus of this chapter is to compare different **solution representations** and their specific **schedulers**. Our results are favorably compared with literature, and we highlight insights for contractors.

In **chapter 5** (Leyman and Vanhoucke, 2016a), we propose a general model for the cash outflows and include capital management for the contractor (**RQ4**). The cash flow distribution model can be seen as a generalized variant of the payment models, which uses

the cash outflows instead of the cash inflows of each activity. The inclusion of capital furthermore states that at no point in time the cash balance, or available capital, can be negative. Hence, cash outflows can only be paid if sufficient capital is available. Both the problem with and without renewable resources are discussed separately, and a different **scheduler** is proposed for each problem. Both schedulers reduce capital shortages by delaying cash outflows such that these cash flows can be compensated by cash inflows of other activities. A clear added value of the capital feasibility method is demonstrated in the results, and three **metaheuristics** with each two variants are implemented to analyze their performance. Finally, we provide managerial insights to contractors with respect to their capital management and the integration with NPV optimization.

In **chapter 6** (Leyman and Vanhoucke, 2016c), the question is posed whether it makes sense to consider the renewable resource availability as a given (**RQ5**). Hence, we assign a cost to each renewable resource and include this cost in the NPV objective, rather than decide on the amount of a resource made available first and schedule the activities second. These resource usage costs are assigned at the start of the project (fixed *timing*), but their *size* depends on the schedule. As a result, resource usage costs are included as a form of variable size of cash outflows in figure 1.3. We design a resource cost reduction step, which is integrated with the **scheduler** of chapter 2. Furthermore, we compare the results with those of the two **solution representations** of chapter 4. Insights are provided in the importance of resource costs and activity cash flows.

Finally, in **chapter 7**, conclusions are drawn from the research presented in this book, and the five research questions are revisited. Recommendations for future research are also discussed in this chapter.

Publications in international journals

- Chapter 2: Leyman, P. & Vanhoucke, M. (2015). A new scheduling technique for the resource–constrained project scheduling problem with discounted cash flows. *International Journal of Production Research*, 53(9): 2771–2786.
- Chapter 3: Leyman, P. & Vanhoucke, M. (2016b). Payment models and net present value optimization for resource-constrained project scheduling. *Computers & Industrial Engineering*, 91: 139–153.
- Chapter 5: Leyman, P. & Vanhoucke, M. (2016a). Capital– and resource–constrained project scheduling with net present value optimization. *European Journal of Operational Research*, Article in press.

Unpublished working papers

- Chapter 4: Leyman, P., Van Driessche, N. & Vanhoucke, M. (2016). Metaheuristics for the discrete time/cost trade–off problem with net present value optimization and different payment models. *Working paper*.
- Chapter 6: Leyman, P. & Vanhoucke, M. (2016c). The resource availability cost problem with net present value objective. *Working paper*.

2

A new scheduling technique for the resource–constrained project scheduling problem with discounted cash flows

In this chapter, we discuss the resource–constrained project scheduling problem with discounted cash flows (RCPSPDC). We introduce a new schedule construction technique which moves sets of activities to improve the project net present value (NPV) and consists of two steps. In particular the inclusion of individual activities into sets, which are then moved together, is crucial in both steps. The first step groups activities based on the predecessors and successors in the project network, and adds these activities to a set based on their finish time and cash flow. The second step on the contrary does so based on the neighboring activities in the schedule, which may but need not include precedence related activities. The proposed scheduling method is implemented in a genetic algorithm (GA) metaheuristic and we employ a penalty function to improve the algorithm's feasibility with respect to a tight deadline. All steps of the proposed solution methodology are tested in detail and an extensive computational experiment shows that our results are competitive with existing work.

2.1 Introduction

The resource–constrained project scheduling problem (RCPSP) has been extensively discussed in literature in the past few decades. Subsequently, the problem has been covered in a multitude of extensions and variations (Hartmann and Briskorn, 2010). Whereas the basic RCPSP and most its extensions focus on total project duration minimization, other variations aim to minimize resource idle time, minimize project costs or maximize the project net present value (NPV).

In this chapter, we focus on the RCPSP with discounted cash flows (RCPSPDC) and aim to maximize a project's total NPV. The RCPSPDC can be seen as a variant of the RCPSP in which each activity has a cash flow, which can be either positive or negative. Furthermore, since the focus is on maximizing the NPV, a deadline is imposed on the project to avoid that activities with negative cash flows may be delayed indefinitely. Figure 2.1 shows that the focus is on the PAC model for both cash in– and outflows in this chapter. From a solution methodology perspective, we develop a new **scheduling technique** and integrate it with two variants of a **metaheuristic**.

	Timing	**Size**	**Timing & size**
Cash in	Payments at activities' completion times (PAC)	Progress payments (PP)	Progress based payment pattern (PBPP)
		Payments at event occurences (PEO)	Expense based payment pattern (PBPP)
Cash out	Payments at activities' completion times (PAC)	Resource usage costs	General capital model

Figure 2.1: Overview of the research on project scheduling with NPV optimization in chapter 2.

The remainder of this chapter is organized as follows. Section 2.2 starts with a literature overview of the existing papers on the RCPSPDC. In section 2.3 we discuss the mathematical problem formulation, whereas in section 2.4 we go into detail about our schedule generation procedure and illustrate all of its steps on a problem example. Section 2.5 gives an overview of our proposed metaheuristic and results of a computational experiment are shown in section 2.6. We finish with a conclusion in section 2.7.

2.2 Literature overview

Overviews of the existing literature on the RCPSPDC and its extensions have been given by Herroelen et al. (1997) and Mika et al. (2005). In this manuscript we briefly discuss all papers to–date which handle the RCPSPDC as formulated in section 2.3. We first discuss the existing exact methods, then the multi–pass heuristics, and finally the metaheuristic procedures.

Exact procedures: Yang et al. (1995) employ a branch–and–bound procedure to solve the RCPSPDC and make use of a depth–first search. The authors apply node fathoming rules to reduce the size of the tree and show that these rules significantly reduce computation times. Icmeli and Erengüç (1996) also propose a branch–and–bound procedure which introduces additional precedence relations to avoid resource conflicts. Branching is done according to the minimal delaying alternatives concept proposed by Demeulemeester and Herroelen (1992) and their results outperform other existing procedures. Another branch–and–bound procedure is used by Baroum and Patterson (1996) and tested on instances from Patterson's dataset, with networks consisting of up to 51 activities. Vanhoucke et al. (2001b) propose a branch–and–bound procedure for the RCPSPDC based on an exact recursive method for the resource–unconstrained case. The procedure can solve relatively small problems to optimality within a limited computation time. Schutt et al. (2012) use lazy clause generation for the RCPSPDC and come up with three appropriate propagators for maximizing the NPV. These propagators are tested using a branch–and–bound algorithm and several binary search heuristics. The authors compare their results with those of Vanhoucke et al. (2001b) and conclude that their proposed method finds both better and a higher number of feasible solutions than those of the benchmark.

Multi–pass heuristics: Russell (1986) proposed the first heuristic for the RCPSPDC and uses six rules to solve the problem. The work has shown that heuristics which perform well for duration minimization do not necessarily provide good results for NPV maximization. A backward scheduling method is used by Smith-Daniels and Aquilano (1987), with a lump–sum payment at activity completion and cash outflows occurring at the start of each activity.

Metaheuristics: Zhu and Padman (1999) apply a tabu search procedure to the RCPSPDC and show considerably better results than any single–pass heuristic available. Kimms (2001) employs Lagragian relaxation and derives tight upper bounds. Based on these upper bounds feasible solutions are constructed, which are shown to be very close to the optimal solutions. Selle and Zimmermann (2003) schedule large–scale projects subject to resource constraints and temporal constraints. A new bi–directional scheduling approach is proposed which simultaneously schedules activities forward and backward. The

new approach manages to outperform existing scheduling approaches. Vanhoucke (2010) employs a scatter search metaheuristic and makes use of a bi–directional schedule generation scheme and a recursive search method to improve the project NPV. The results are compared with both the exact procedure of Vanhoucke et al. (2001b) and several different metaheuristics coded by the author, including the genetic algorithm of Vanhoucke (2009). It is concluded that the proposed method outperforms all others. Next, Gu et al. (2012) discuss the RCPSPDC for large project instances. The authors apply Lagrangian relaxation to projects with up to 11,000 activities and employ three improvements to ensure scalability of their algorithm. These steps involve the relaxation of precedence constraints, parallel implementation and a hierarchical subgradient algorithm. The results produced are highly competitive given a reasonable computation time. Gu et al. (2013) improve on the results of Schutt et al. (2012) by making use of a Lagrangian relaxation based forward–backward improvement heuristic, to ensure tight deadlines are met. The authors' forward–backward method used is that of Li and Willis (1992), who use the activity start (finish) times of the previously generated forward (backward) schedule to construct the current schedule. The authors compare their procedure with the algorithm of Vanhoucke (2010) based on a 5 minutes time limit and find that their method performs best. It is however important to note that the results of Vanhoucke (2010) are based on a 5,000 schedule limit instead of a run time restriction and have an average computation time of 2.2 seconds.

Based on this overview, we conclude that the papers of Vanhoucke (2010) and of Gu et al. (2013) are the most recent ones which discuss a metaheuritic solution methodology for the RCPSPDC. Thus, we will compare the results of our algorithm with the procedures of these papers in section 2.6.

2.3 Problem formulation

A project can be represented as an activity–on–the–node (AoN) network $G(N, A)$ with N representing the nodes or project activities, and A the network arcs or precedence relations between the activities. For the precedence relations a time–lag of zero is assumed. The activities are numbered from the start dummy 0 to the end dummy $n + 1$. Each activity i ($i \in N = \{1, \ldots, n\}$) has a duration d_i, renewable resource demand r_{ik} and cash flow c_i. The latter is composed by discounting the pre-specified cash flow cf_{it} of an activity i at time t to its finish time and can be mathematically formulated as follows: $c_i = \sum_{t=1}^{d_i} cf_{it} \cdot e^{\alpha(d_i - t)}$. Each renewable resource k has a limited constant availability of a_k. The decision variables f_i contain the finish time for each activity i. Finally, the project has a deadline δ_{n+1}.

The RCPSPDC was proven to be NP–hard by Blazewicz et al. (1983) and can be represented as $m, 1|cpm, \delta_n, c_i|npv$ according to the classification scheme of Herroelen et al. (1999), and as $PS|prec|\sum C_i^F \beta^{C_i}$ according to Brucker et al. (1999).

Mathematically, the problem is conceptually formulated as follows:

$$\text{Maximize} \sum_{i=1}^{n} c_i \cdot e^{-\alpha f_i} \tag{2.1}$$

Subject to:

$$f_i \leq f_j - d_j, \quad \forall (i,j) \in A, \tag{2.2}$$

$$\sum_{i \in S(t)} r_{ik} \leq a_k, \quad k = 1, \ldots, R; \; t = 1, \ldots, \delta_{n+1}, \tag{2.3}$$

$$f_{n+1} \leq \delta_{n+1}, \tag{2.4}$$

$$f_i \text{ integer}, \quad \forall i \in N \tag{2.5}$$

The objective function (2.1) aims to optimize the project NPV, whereas constraints (2.2) ensure precedence feasibility. Constraints (2.3) impose the renewable resource limits with $S(t)$ the set of activities in progress at time t. Constraint (2.4) enforces deadline feasibility and finally constraints (2.5) ensure the decision variables are integer.

2.4 Schedule generation

In this section, we discuss the schedule generation employed by the metaheuristic (section 2.5) and all of the included steps. A distinction is made between two variants of the scheduling approach based on the percentage of activities with a negative cash flow. If this percentage is lower than or equal to 50%, we start with a forward schedule generation scheme (SGS) and subsequently delay activities. If more than 50% of the activities have a negative cash flow, a backward SGS is first applied followed by the advancing of activities. The overall flow of the schedule generation can be seen in figure 2.2. The subsequent subsections go into detail about each individual step of the scheduling process.

2.4.1 Initial schedule and deadline feasibility

The first step in the scheduling process is the construction of an initial deadline–feasible schedule. To construct this schedule we use the well–known forward serial schedule generation scheme (SSGS) (Kelley, 1963; Kolisch, 1996) if no more than half of the activities have a negative cash flow. If the result is feasible with respect to the project deadline, we continue with the first set of activity move rules as described in section 2.4.2. If however,

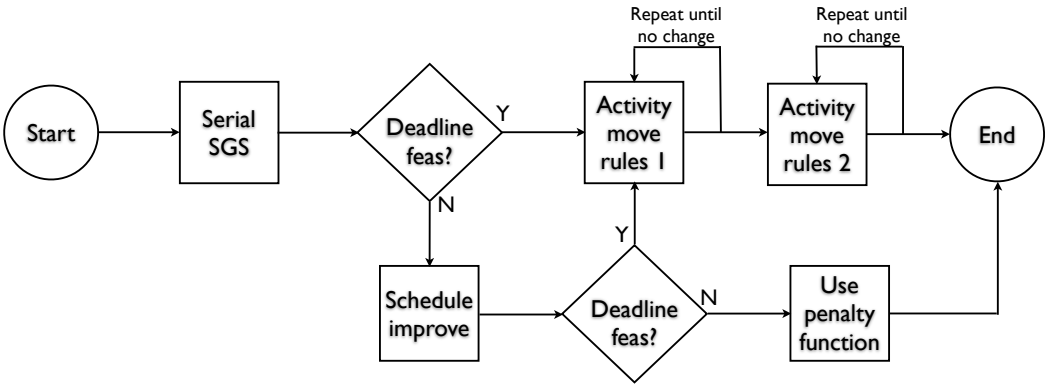


Figure 2.2: Schedule generation flow.

the schedule is infeasible (*D–Infeas*) we apply the forward–backward improvement method of Li and Willis (1992) to reduce the project duration, until no further improvement is possible. If the resulting schedule after applying Li and Willis (1992) is feasible, we proceed with the activity move rules. If even after the schedule improvement method the schedule is still infeasible, we add the following penalty function to the project NPV:

$$\begin{cases} NPV = -Y_1 + NPV_{infeas} \cdot Y_2^{f_{n+1}-\delta_{n+1}} & \text{if } NPV_{infeas} \geq 0 \\ NPV = -Y_1 + \frac{NPV_{infeas}}{Y_2^{f_{n+1}-\delta_{n+1}}} & \text{otherwise} \end{cases} \tag{2.6}$$

with Y_1 and Y_2 variables to be tested in section 2.6.1, and NPV_{Infeas} the NPV of the *D–Infeas* schedule.

- Y_1 represents a large fixed value which is subtracted from the project NPV to ensure that the infeasible solution's NPV is considerably worse than that of any feasible solution.
- Y_2 aims to penalize the NPV based on the difference between the solution's project duration and the project deadline. As such this parameter has a fractional value in order to exponentially reduce the project's NPV. The value of the parameter furthermore depends on the absolute value of the NPV, with a higher absolute NPV requiring a parameter value closer to 1 than a lower absolute NPV.

As an example, assume a project with a total duration of 19, a project NPV of 800 and a deadline of 17. If $Y_2 = 0.9$ we first adjust the NPV by multiplying 800 with 0.9^{19-17} and the remaining NPV is 648. We then subtract Y_1 from this value, assume $Y_1 = 1,000$, and receive -352. This way the NPV of the *D–Infeas* project has been reduced from 800 to -352. If by comparison the project duration is 18 instead of 19, the adjusted NPV would

be -280. As such, a schedule with a lower deadline violation is penalized less than one with a higher deadline violation. Both are however reduced to a value well below that of NPV_{Infeas}.

If the project NPV is on the contrary relatively small, assume 80, and the deadline is again 17, the corrected NPV would be -935.20 for a project duration of 19 and -928 for a duration of 18. In this case the absolute difference between both corrected values is much smaller than it was for a larger project NPV (7.2 versus 72). If we however apply a lower value for Y_2, e.g. 0.50, the corrected NPV becomes -980 and -960 respectively. As a result, the difference between both corrected NPV has increased (20 versus 7.2), more clearly distinguishing both schedules from one another.

In case more than half of the activities have a negative cash flow, the backward instead of the forward scheduling approach is selected. If the resulting schedule proves to be infeasible with respect to the project deadline, we again apply the method of Li and Willis (1992) until no improvement can be found. The schedule is then once more evaluated. If it is still *D–Infeas* we use the penalty function, otherwise we move on to the activity move rules.

2.4.2 Activity move rules

This subsection gives an overview of the rules applied to delay (advance) activities, starting from the initial forward (backward) schedule constructed by the SSGS. In order to illustrate these rules, we use an example which is shown at the top of figure 2.3. We assume a single renewable resource with an availability of 5 and a project deadline of 22. The initial schedule, based on the forward SSGS ($\leq$50% negative cash flows), is shown at the bottom of figure 2.3 with the renewable resource (RR) on the vertical axis and the time on the horizontal axis. The activities are scheduled according to the priority list (PL) (1, 2, 3, 4, 6, 7, 5, 8, 9, 10). The initial NPV based on a discount rate of 1% is 25.72 (= 38.82 + 19.22 - 28.82 + 13.85 - 4.30 + 4.62 - 17.56 - 8.61 + 25.06 - 16.54). Note that the schedule shown is deadline–feasible, and no schedule improvement is needed. Finally, since we start from a forward schedule, we aim to delay sets of activities.

2.4.2.1 Network–based moves

The first set of rules delays or advances activities based on the precedence relations in the project network. Depending on whether the initial schedule is an earliest finish (forward scheduling) or latest finish (backward scheduling) schedule, the procedure aims to delay or advance activities. First, we discuss the delay algorithm in detail and then we highlight the differences when activities should be advanced.

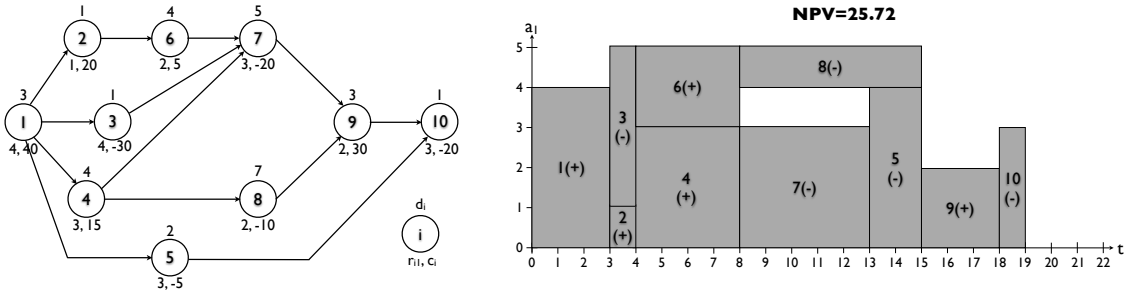


Figure 2.3: Network & initial schedule example 1.

With an initial forward schedule, we delay both individual activities with a negative cash flows and sets of activities with a negative cumulative NPV (NPV_{cum}). Starting from the last activity in the PL each activity is evaluated with respect to its cash flow. If the activity cash flow c_i is negative, the activity finish time is not equal to its latest finish time lf_i and a delay is possible based on the successors, the algorithm delays the activity as much is possible. If however a delay is impossible due to at least one successor with a start time equal to the activity's finish time, algorithm 1 is applied. This recursive method includes all activities which should be delayed together in a set. Starting from the current activity it adds all successors S_i which start immediately after this activity and then recursively calls the procedure again for each of these successors. For each of those, all predecessor activities P_i with both a negative cash flow and a finish time not smaller than the initial activity's finish time are also added to the activity set. Note that algorithm 1 does not take the predecessors of the start activity into account, because these will be considered in a next iteration. Once this set is complete and algorithm 1 returns to the start activity, the cumulative NPV of the set is calculated and evaluated. If it is negative, the minimum delay is calculated based on the earliest successor not in the set, and this for all activities in the set. The global minimum of all these minima then constitutes the maximum allowable delay and can be formulated as follows: $\Delta = min\{f_k - d_k - f_i | i \in setAct, k \in S_i, k \notin setAct\}$. Then each activity in the set has its finish time increased by this Δ. If the schedule is feasible with respect to the renewable resources the current finish times are retained. If the schedule is infeasible we decrease Δ by 1 and continue until we find a feasible solution or until Δ reaches 0. When the search for a delay is complete, the procedure moves on the previous activity in the PL. Finally, the algorithm is repeated until no more changes occur and the procedure reaches the first activity in the PL without delaying any activity.

If the initial schedule is a backward one, the goal is to advance both individual activities with a positive cash flow and sets of activities with a positive cumulative NPV. This states

Algorithm 1 Get all successors

GetAllSuc (current activity i, start activity k, $set[]$, NPV_{cum})
  $\forall j \in S_i$
    If $j \notin set \wedge f_j - d_j = f_i$
      Add j to set
      Add NPV_j to NPV_{cum}
      GetAllSuc (j, k, set, NPV_{cum})
    End if
  $\forall j \in P_i$
    If $j \notin set \wedge f_j + d_i = f_i \wedge j \neq k \wedge f_j \geq f_k \wedge f_j < lf_j \wedge c_j < 0$
      Add j to set
      Add NPV_j to NPV_{cum}
      GetAllSuc (j, k, set, NPV_{cum})
    End if
  Return set, NPV_{cum}

the first major difference with the delay procedure, where we aimed to delay activities. As a consequence, we now have to consider the finish time of predecessors instead of the start time of successors when determining the potential change in activity finish time. Finally, algorithm 1 needs to be inverted to a **GetAllPred** variant which finds all predecessors of an activity and any successors with both a positive cash flow and a finish time no more than that of the start activity.

It is important to note that at most half of the activities are considered for a move. Recall that we start from a forward (backward) schedule if at most (less than) half of the activities has a negative cash flow meaning that at most half of the activities has to be considered for a delay (advance).

To illustrate these rules we go back to the example. Recall that the PL of the example was (1, 2, 3, 4, 6, 7, 5, 8, 9, 10). Starting from the final activity, we first consider activity 10. Since it has a negative cash flow and no successors it is delayed until time 22. Next is activity 9 which however has a positive cash flow and is skipped. We move on to activity 8 which has a negative cash flow but cannot be delayed due to its successor 9. Algorithm 1 is used and returns the set {8, 9} since 9 is the only successor of 8. No other activities are included since 9's only other predecessor is 7, but 7 finishes 2 time units before the start of 9. The cumulative NPV of both 8 and 9 is positive so they are not considered for delay. To further illustrate why only 8 and 9 are in the set, consider the top left of figure 2.4 which shows the project network but with only those precedence relations which constitute equalities in the mathematical model of section 2.3. In the figure activity 9 is only connected to 8, which still has 4 as a predecessor, but this activity is not taken into account. This way the set is limited to the activities 8 and 9. Next in the PL is activity 5 which has a negative cash flow and whose only successor is 10. As such, activity 5 can be delayed beyond activity 9 to time 21. Note that due to this delay, the ordering of the

activities in the schedule is changed. After 5 also activity 7 can be delayed because its only successor is 9 which is scheduled later. Activities 6 and 4 are not delayed because they have a positive cash flow, whereas 3 cannot be delayed due to the renewable resource constraint. Finally, activities 2 and 1 are also not eligible for delay because of their positive cash flow.

Since at least one delay has occurred, the procedure starts again. Activities 10 and 9 can be skipped because the former is scheduled at its latest finish time and the latter has a positive cash flow. Next is activity 8 which cannot be delayed because of its successor 9. If algorithm 1 is applied this time, it returns the set $\{7, 8, 9\}$ because 7 as a predecessor of 9 now has a finish time equal to its successor's start time. The cumulative NPV of the three activities is negative and their maximum allowable delay is 1. Since this delay is feasible with respect to the renewable resource, the three activities are all delayed by 1 time unit. The top right graph of figure 2.4 illustrates that activity 7 is now also added to the set because its finish time equals activity 9's start time and hence both are connected. No further delays are possible both in this iteration and the procedure's next, so it terminates. The resulting schedule can be found at the bottom of figure 2.4.

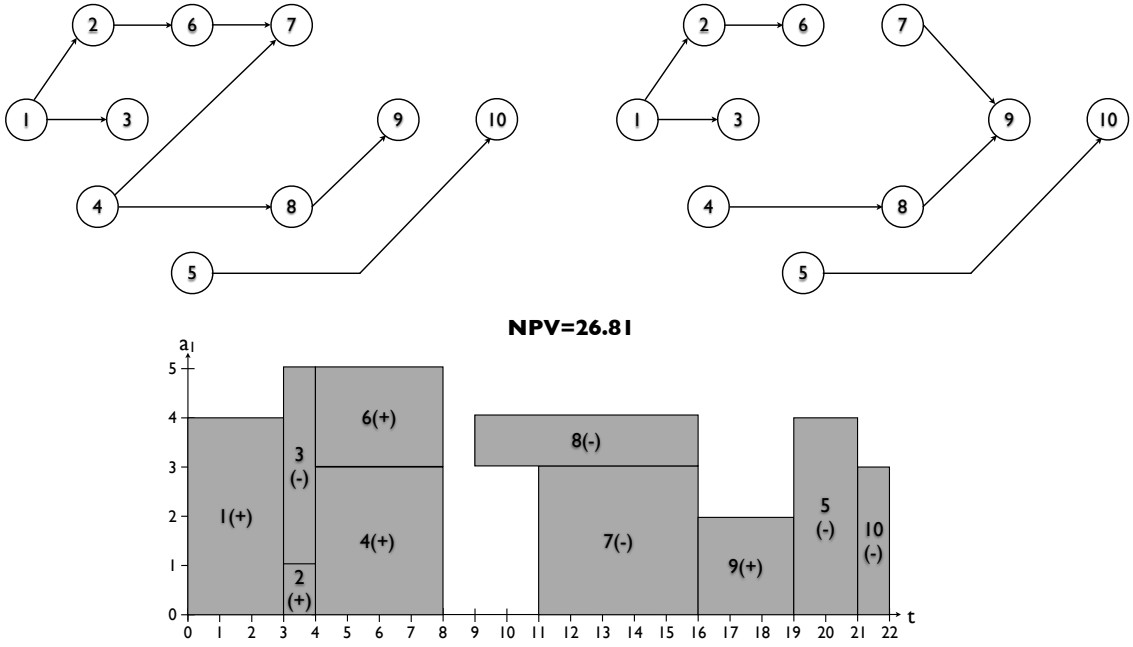


Figure 2.4: Network–based delays example 1.

2.4.2.2 Schedule–based delays

The second set of rules delays or advances activities based on neighboring activities in the project schedule, which can but need not be precedence related. Whereas in section 2.4.2.1 activities were grouped together based on their precedence relations, we here group activities together based on their neighbors, which means that one activity's finish time equals another's start time. These neighboring activities may be precedence related, but can also be scheduled one after another because of the renewable resources. This way, the first set of rules is extended, because these rules may not delay activities due to a resource conflict with other activities. In such cases, it may however be possible that if all of these activities would be considered together, in spite of the absence of any precedence relations between some of them, a delay would be beneficial. Algorithm 2 shows how to find all such neighboring activities. The manner in which delays occur and with what Δ are the same as for the network–based delays in section 2.4.2.1.

Algorithm 2 Get all later neighbors

GetLaterNeighb (current activity i, start activity k, $set[]$, NPV_{cum})
  $\forall j \in N \wedge f_i = f_j - d_j$
    If $j \notin set$
      Add j to set
      Add NPV_j to NPV_{cum}
      GetLaterNeighb (j, k, set, NPV_{cum})
    End if
  $\forall j \in N \wedge f_i - d_i = f_j$
    If $j \notin set \wedge j \neq k \wedge f_j \geq f_k \wedge f_j < lf_j \wedge c_j < 0$
      Add j to set
      Add NPV_j to NPV_{cum}
      GetLaterNeighb (j, k, set, NPV_{cum})
    End if
  Return set, NPV_{cum}

Just like for the first set of rules, we distinguish between a delay and an advance case. The differences between a forward and backward approach are however the same as those discussed in section 2.4.2.1 and are hence not repeated here. Obviously, in this case we also need a reverse version of algorithm 2 to find an activity's earlier neighbors.

Also, similar to the first set of rules, only at most half of the activities are considered for a move.

Let us apply this second set of rules to the example. Starting from the back of the PL and the schedule of figure 2.4 it should be clear that activities 10, 9, 8, 5 and 7 should not be considered anymore since they cannot be delayed any further. The only remaining activity with a negative cash flow is 3, but as stated in section 2.4.2.1 there are insufficient renewable resources available to allow for a delay. This means that the first set of rules will never delay activity 3 because none of its successors have a start time equal to activity

3's finish time. If we however apply algorithm 2 we notice that both activities 4 and 6 have a start time equal to 3's finish time, hence both are added to the activity set. This way, the resulting set is {3, 4, 6} and has a negative cumulative NPV. The maximum allowable delay is 1, because of activity 8 as a successor of 4, and is feasible with respect to the renewable resource. To further illustrate why exactly these activities are included in the set, consider both graphs at the top of figure 2.5 which show the neighbors of each activity both before and after the delay.

Although the procedure for this second set of rules is repeated because a change has occurred, no further improvements are possible. The bottom of figure 2.5 shows the example schedule after the second set of rules has been applied, which is also the optimal one.

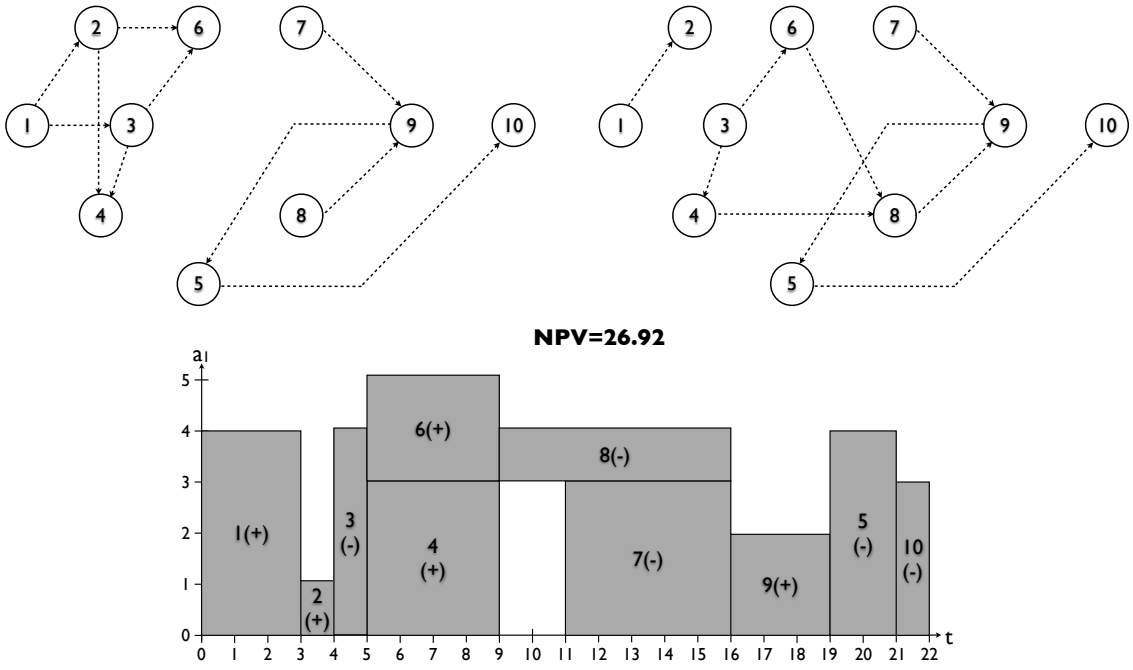


Figure 2.5: Schedule–based delays example 1.

Note that for both sets of activity move rules, the activities are considered starting from the last activity in the PL. This is done to allow the enveloping metaheuristic (section 2.5) to fully exploit its potential. Let us illustrate the potential problems of not using the discussed approach, but for instance start with the activities which have the *smallest cumulative NPV*, based on the example network in figure 2.6. The project deadline is 15, the renewable resource availability is 5 and the discount rate is again 1%. If we start from an initial schedule with finish times {0, 1, 5, 9, 10, 10} then it should be clear that improvement is possible, as both activities 4 and 5 have a negative cash flow and are

only succeeded by the dummy end activity. The optimal solution in this case is to delay activity 5 to time 15, and delay 4 to 14, as can be seen from the bottom left schedule in figure 2.6. If we would however employ a heuristic such as the *smallest cumulative NPV first* rule, activity 4 will always be delayed first, as can be seen in the bottom right schedule in figure 2.6. What is worse is that regardless of the activity ordering, meaning that regardless of which one of both activities has the highest priority and occurs after the other in the PL, activity 4 will be moved first. This means that not only will our solution always be suboptimal, it will also always be the same independent of the PL provided by the metaheuristic. Thus, we have chosen to consider activities for delay in the backward order of the PL, rather than a static heuristic like the one discussed here. Initial tests with heuristics such as *smallest cumulative NPV first* and *largest finish time first* confirmed this reasoning.

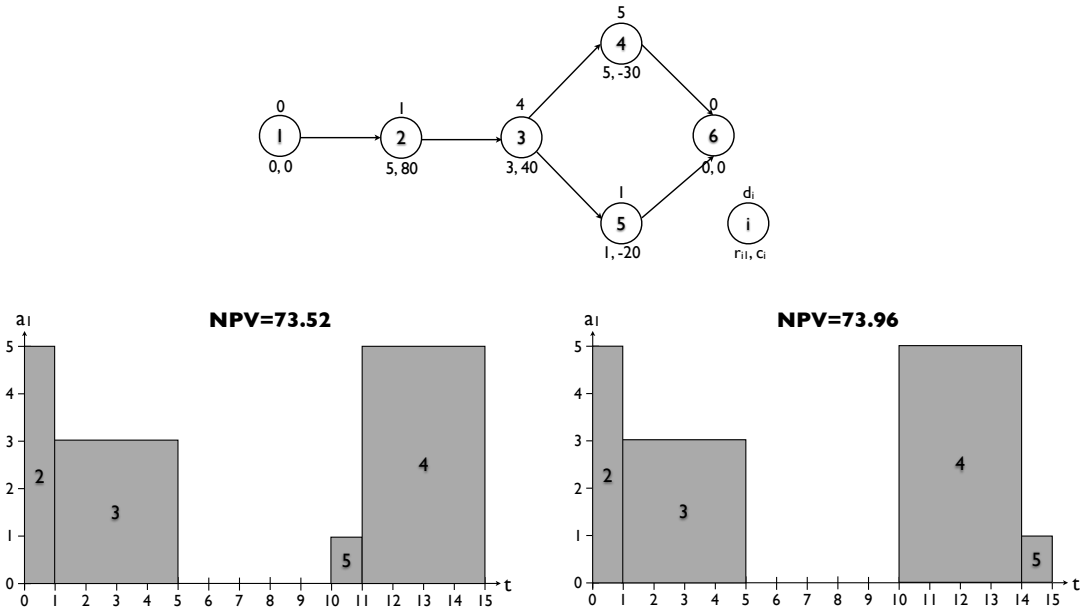


Figure 2.6: Network, optimal and suboptimal schedule example 2.

2.5 Genetic algorithm

Whereas in the previous section the focus was on the schedule generation of a solution, we here discuss the proposed metaheuristic, namely a genetic algorithm (GA).

The GA was first proposed by Holland (1975) and is inspired by evolutionary biology. The technique makes use of such operators as selection, crossover and mutation to combine existing solution into new ones, and has already extensively been used in existing

project scheduling literature. A GA typically has a selection operator which selects parent elements to be combined in new offspring using a crossover operator and which constitutes the intensification step of the algorithm. A small percentage of these children are then mutated to diversify the population. Finally, a population update is applied to reduce the population size by retaining the best parents and replacing the rest by the best children.

We continue this section by going into detail about the selected solution representation, and then discuss each part of our GA in more detail. The differences between two options of our GA are discussed in detail in the following subsections.

2.5.1 Representation

In section 2.4.2.2 we emphasized the importance of delaying activities in the order of the solution's PL. We can further improve upon this by using the topological order (TO) instead of the PL representation (Valls et al., 2004, 2003). The TO representation as used by Debels et al. (2006) not only ensures the ordering of activities is precedence feasible, but also takes the actual activity start or finish time into account. Once all activity move rules have been applied to a schedule, its corresponding PL is transformed into a finish time ordered activity list. Finally, if two activities have the same finish time, we break ties by randomly selecting one of them to go first in the TO representation.

To illustrate the TO we take another look at the first example. Recall that there, we started with the PL (1, 2, 3, 4, 6, 7, 5, 8, 9, 10). Based on the final schedule of figure 2.5 the TO representation becomes (1, 2, 3, 6, 4, 7, 8, 9, 5, 10), with 6 randomly chosen from {4, 6} as tie–breaker to go earlier in the list since both have the same finish time, and 7 randomly chosen from {7, 8}.

2.5.2 Initial population

The initial population is generated by creating random precedence feasible PLs for each element in the population. Whereas typically the initial population is larger than the actual population, our tests indicated that better results are achieved by simply copying the initial population as the start population of the metaheuristic. This initial population generation is the same for both GA1 and GA2.

2.5.3 Selection

For the selection operator of our GA we have chosen to implement two alternatives. Selection 1 for GA1 constitutes a four–tournament selection for both parents. This implies that for both the father and mother four elements are randomly selected from the population, of which in both cases the best one is retained.

In the case of GA2 we employ an elite selection, which means that the father is always selected out the pool of best $|X_2|$ elements. This does however not mean that the set of best X_2 elements is always the same. Once the population as a whole has been updated, the set with size $|X_2|$ is updated as well since the new elements may contain better solutions. Each element in the population is evaluated with respect to its NPV and the best $|X_2|$ elements are stored as an elite set. This way this subset always contains the best $|X_2|$ elements once a population update has been done. The mother on the contrary is still chosen using a four–tournament selection.

Other selection operators for both parents in GA1 and the mother in GA2, such as a roulette wheel and rank selection were tested, but the selection discussed earlier performed best.

2.5.4 Crossover

Our crossover operator is a one–point crossover as typically used in a GA, for both GA1 and GA2. We also tested the two–point crossover, the MCUOX (Ulusoy et al., 2001), and the partially–mapped crossover, but they were all outperformed by the one–point crossover.

2.5.5 Mutation

For both GA1 and GA2 we use the same mutation operator, namely a two–activity swap. The difference between both algorithms however is the employed mutation rates R_1 and R_2. For GA1 we expect that a relatively low mutation rate is sufficient since its methods correspond with those typically used in literature. GA2 on the other hand requires a higher rate to serve as counterweight to the elite selection of section 2.5.3 to ensure the population is diverse enough. Our results of section 2.6.1 confirm these assumptions.

Alternatively, the mutation operators scramble, inversion and insertion were tested, but they performed worse than the proposed swap operator.

2.5.6 Evaluation and population update

For updating the population we retain the best $|X_1|$ parents for GA1 and best $|X_2|$ for GA2. The rest of the parents are replaced by the best newly generated children. Note that the X_2 parameter also determines the size of the pool out of which the father is always selected in GA2.

2.6 Computational results

In this section, we show and discuss our computational results by first configuring both GAs, and then by comparing with the best known results from literature.

In terms of test data, we have chosen to use the same projects as employed by Vanhoucke (2010), downloaded from *www.projectmanagement.ugent.be*. On this site the author's solutions, computation times and employed upper bounds are also available for each datafile. The dataset itself consists of 720 networks with 25, 50, 75 or 100 activities. Each datafile can be executed with 6 different cash flow files, with the percentage of negative cash flows (*%Neg*) ranging from 0% to 100%, in steps of 20%. Furthermore, a deadline is imposed based on the optimal RCPSP project duration of the procedure by Demeulemeester and Herroelen (1992) for the projects with 25 activities and on the best known heuristic solutions of Debels and Vanhoucke (2007) for a higher number of activities. This minimum project duration is then increased by a specific percentage (*D–Incr*), ranging from 5% to 20% in steps of 5%, and constitutes the project deadline. As such, the problem set contains 720 * 6 * 4 = 17,280 problem instances. The order strength (*OS*) and resource constrainedness (*RC*) of the networks are both either 0.25, 0.50 or 0.75. The resource usage (*RU*) is 2 or 4. Finally, we employ a discount rate of 1%. A summary of the parameters of the dataset is given in table 2.1.

Parameter	**Values**
Act	25, 50, 75, 100
OS	0.25, 0.50, 0.75
RU	2, 4
RC	0.25, 0.50, 0.75
D–Incr	5, 10, 15, 20
%Neg	0, 20, 40, 60, 80, 100

Table 2.1: Parameters dataset.

2.6.1 Configuration of the algorithm

In this subsection, we configure both our GAs. We first give an overview of the values of the parameters of both algorithms and of the penalty function. Then we show that the proposed order in which the two sets of activity move rules are applied is the best and illustrate the relevance of both steps. Finally, we compare both GAs and also show the results if 5,000 random schedules had been generated instead of making use of the proposed metaheuristic. This way the added value of our GA approaches is also emphasized. Each time we make use of the 5,000 schedules termination criterion, and employ 20% of data.

The parameters that were tested are the population sizes $|P_1|$ and $|P_2|$, the number of retained elements $|X_1|$ and $|X_2|$ and the mutation rates R_1 and R_2. Computational experiments showed a best $|P_1|$ and $|P_2|$ equal to 50, $|X_1|$ equal to 10 and $|X_2|$ equal to 5, and a mutation rate R_1 of 20% and R_2 of 95%. Note that the assumption of GA2 needing a higher mutation rate R_2 than R_1 of GA1 as stated in section 2.5.5 is confirmed. The tested population sizes range from 25 to 100 in steps of 5, whereas the number of retained elements $|X_1|$ and $|X_2|$ tested range from 1 to 15 in steps of 1. Finally, the mutation rates were tested in steps of 5%, from 5% to 95%.

The Y_1 and Y_2 parameters of the penalty function have also been tested.

- The tested values for Y_1 range from $15,000$ to $25,000$ in steps of $1,000$ and its optimal value equals $20,000$, much larger than that of any feasible solution's NPV.
- Y_2 was initially tested with values between 0.50 and 0.95 in steps of 0.05. Additional finetuning of the latter was done to further improve the performance of the penalty function. The parameter's optimal value was found to depend on the absolute value of the average NPV ($AvNPV$) of all test instances with a fixed percentage of negative cash flows. The values for Y_2 are found in table 2.2. It should be clear based on the table that the suggested relation does indeed exist. As the absolute value of $AvNPV$ decreases, so does the factor Y_2. Observe in table 2.2 that starting from 0% negative cash flows Y_2 decreases along with the absolute average NPV until 60%, whereas from 80% on both increase together. This way, our results confirmed what we already suspected in section 2.4.1, namely that a lower Y_2 value is required for a lower absolute project NPV, to more clearly distinguish solutions with a different deadline violation from one another.

$\%Neg$	0%	20%	40%	60%	80%	100%
$AvNPV$	7,930.42	4,763.23	1,436.95	263.86	-1,369.12	-5,736.01
Y_2	0.995	0.95	0.85	0.70	0.92	0.97

Table 2.2: Parameters of the penalty function.

Next, we compare different combinations of the activity move rules of section 2.4 and use GA2 to make this comparison. The reason that GA2 and not GA1 is used for the comparison, is because GA2 not only performs better, but also because it more clearly illustrates the value of each step. To validate both distinct steps, each step is first excluded from the method. These results are displayed in the table under *NoStep1* (the network–based moves are excluded) and *NoStep2* (the schedule–based moves are excluded). Finally, we also tested a switch in the order in which the network– and schedule–based moves are

applied (*Switch1&2*). Recall that otherwise we first apply step 1 and then step 2, hence for the *Switch1&2* case we first apply step 2 and then step 1. As a benchmark, we use the case in which only single activity moves are applied (*OnlySingle*).

The results of this analysis are shown in table 2.3 and are based on their deviations from the best known solutions for the resource–unconstrained max–NPV problem, as discussed by Vanhoucke (2006). The table displays the relative average deviation (*%AvDev*) which stands for the percentage average deviation of the solutions of this manuscript in comparison with the results for the resource–unconstrained case. A lower value corresponds with a better solution. The results from the comparison with the max–NPV in the table only take those data files into account for which all alternatives are able to find deadline feasible solutions, this to ensure a correct comparison. The table also compares all options based on the percentage average difference (*%AvDiff*), which shows the performance of the proposed method in comparison with the scatter search of Vanhoucke (2010). A positive value corresponds with a better performance of the proposed option, whereas a negative value means that the scatter search has better results. Finally, the percentage of better solutions (*%Better*) found by each combination in comparison with the scatter search is shown.

Based on the results in the table it can be seen that *BothSteps* does better than all other options on all three performance criteria, which means that the proposed scheduling steps and their ordering are justified.

Option	BothSteps	NoStep1	NoStep2	Switch1&2	OnlySingle
%AvDev	**211.37**	214.55	214.29	214.73	216.60
%AvDiff	**10.67**	8.68	8.72	8.87	3.95
%Better	**60.48**	59.50	58.08	59.53	53.53

Table 2.3: Comparison of schedule step combinations.

Finally, table 2.4 compares GA1 and GA2 with 5,000 random schedules, only problem instances for which all three methods could find a deadline feasible solution are taken into account. The percentage feasible (*%Feas*) shows the number of deadline feasible solutions found.

Based on the table the following can be concluded:

- GA2 performs best both overall and for all values of the six parameters of the dataset. Hence, in section 2.6.2 where we compare with existing methods, we only show the results of GA2.

- GA2 furthermore reports the largest percentage of feasible solutions found. Additionally, the percentage of feasible solutions found between on the one hand both

GAs and the 5,000 random schedules on the other hand illustrates the added value of the proposed penalty function as part of our metaheuristic.

- The difference in performance between GA2 and GA1 in terms of *%Neg* is larger when *%Neg*$>$ 50% compared to the cases with *%Neg*$\leq$ 50%.
- The larger the *OS* the better GA2 performs in comparison with GA1.
- The smaller the *RC* the better GA2 performs in comparison with GA1.

		GA1		**GA2**		**5,000 randoms**	
		%AvDev	*%Feas*	*%AvDev*	*%Feas*	*%AvDev*	*%Feas*
Act	25	198.93	99.44	194.41	99.51	235.53	98.56
	50	319.23	98.84	305.53	99.03	412.73	89.77
	75	151.73	97.18	148.71	98.06	200.35	79.70
	100	183.76	96.64	177.89	97.55	230.93	74.10
OS	0.25	147.94	96.82	143.68	98.00	201.63	83.07
	0.50	213.97	97.66	208.39	97.95	265.62	83.02
	0.75	280.69	99.60	270.24	99.65	344.55	90.50
RU	2	271.19	97.36	263.06	98.38	344.73	81.16
	4	166.71	98.69	160.97	98.69	208.10	89.91
RC	0.25	368.35	97.38	357.92	98.13	468.72	84.55
	0.50	137.72	98.28	132.20	98.75	172.02	85.07
	0.75	145.22	98.42	140.48	98.73	181.19	86.98
D–Incr	5%	251.73	92.78	248.68	94.21	306.27	61.13
	10%	124.45	99.81	114.68	99.93	154.02	84.19
	15%	344.50	99.51	335.10	100.00	441.55	96.97
	20%	147.74	100.00	143.43	100.00	189.31	99.84
%Neg	0%	30.77	98.33	30.53	98.85	32.87	91.98
	20%	28.49	98.75	28.02	99.34	32.60	92.22
	40%	73.57	98.99	72.53	98.85	92.47	93.36
	60%	572.16	97.36	551.35	97.40	743.61	78.72
	80%	402.13	97.60	388.02	98.33	518.50	79.10
	100%	277.84	97.12	270.08	98.44	329.13	78.82
Overall		216.21	98.03	209.34	98.54	272.84	85.53

Table 2.4: Comparison between GA1, GA2 and 5,000 randoms.

2.6.2 Comparison with literature

To the best of our knowledge, the scatter search of Vanhoucke (2010) and the Lagrangian–based heuristic of Gu et al. (2013) are the best known algorithms for the RCPSPDC. Both however report their results based on a different termination criterion. Whereas Vanhoucke (2010) uses the 5,000 schedules criterion, Gu et al. (2013) terminate after 5 minutes. As such we compare with both algorithms separately.

First, in table 2.5 we compare our GA with Vanhoucke (2010). The percentage of instances for which the best known solution is found (*%Best*) constitutes the number of instances for which either method found the best known solutions. Note that the sum of both percentages is larger than 100% because of the cases in which both algorithms have the same result are included twice.

The following conclusions can be drawn from the table:

- Overall our results outperform those of Vanhoucke (2010).
- Our proposed method performs better comparatively with a decreasing value of the *OS*, and an increasing value of the *RC* and *RU*.
- A similar trend can be observed as *Act* increases, but it is less pronounced.
- For the instances with *%Neg* equal to 40% or 60% the *%AvDev* of GA2 is worse than that of Vanhoucke (2010), but this is offset by a larger *%Best*.

		This chapter			**Vanhoucke (2010)**		
		AvDev	*%AvDev*	*%Feas*	*AvDev*	*%AvDev*	*%Feas*
Act	25	371.41	192.59	99.51	370.89	192.46	100.00
	50	1,295.24	294.05	99.03	1,327.73	291.75	99.86
	75	2,458.97	153.97	98.06	2,540.08	160.30	99.72
	100	3,744.92	204.38	97.55	3,858.48	223.42	99.58
OS	0.25	2,254.87	166.00	98.00	2,356.44	179.67	99.69
	0.50	2,046.98	203.56	97.95	2,098.45	210.55	99.69
	0.75	1,577.06	263.71	99.65	1,593.62	260.23	100.00
RU	2	1,696.00	251.76	98.38	1,746.26	252.45	99.79
	4	2,217.88	171.18	98.69	2,280.20	181.82	99.79
RC	0.25	1,887.70	342.07	98.13	1,942.33	342.11	99.58
	0.50	1,983.81	141.25	98.75	2,039.25	149.67	99.90
	0.75	2,000.15	151.81	98.73	2,058.96	160.33	99.90
D–Incr	5%	2,119.16	228.94	94.21	2,192.98	234.86	99.17
	10%	1,970.61	133.91	99.93	2,052.91	152.41	100.00
	15%	1,919.04	337.93	100.00	1,945.71	332.34	100.00
	20%	1,830.39	145.83	100.00	1,873.86	149.71	100.00
%Neg	0%	4,480.55	30.84	98.85	4,535.76	31.16	99.79
	20%	2,569.39	28.59	99.34	2,626.96	29.17	99.79
	40%	1,022.16	71.21	98.85	1,064,49	69.78	99.79
	60%	703.17	487.85	97.40	758.32	482.66	99.79
	80%	933.29	415.22	98.33	1,023.84	442.50	99.79
	100%	2,011.40	240.45	98.44	2,048.43	252.93	99.79
Overall		1,957.38	211.40	98.54	2,013.68	217.08	99.79

Table 2.5: Comparative computational results part 1 (5,000 schedules).

Table 2.6 shows a more detailed overview of the comparative performance of both algorithms. In this table both algorithms are compared based on the percentage average difference (*%AvDiff*) between them, which shows the performance of the proposed method in comparison with the scatter search of Vanhoucke (2010). A positive value corresponds with a better performance of our GA2, whereas a negative value means that the scatter search has better results. The percentages of better, equal and worse (*%Better*, *%Equal* and *%Worse*) solutions by GA2 in comparison with the scatter search are also shown. Again, only problem instances for which both methods could find a deadline feasible solution are taken into account in both tables. Based on both table 2.5 and 2.6, it can be seen that GA2 outperforms the algorithm of Vanhoucke (2010). The only case for which a slightly worse performance can be observed based on all comparative factors is when *Act* is set to 25.

		%AvDiff	*%Better*	*%Equal*	*%Worse*
Act	25	-1.35	30.03	32.43	37.54
	50	14.52	66.99	1.50	31.51
	75	16.97	73.68	0.24	26.09
	100	12.69	71.67	0.00	28.33
OS	0.25	12.90	70.01	3.29	26.70
	0.50	9.49	60.85	7.44	31.71
	0.75	9.63	50.75	15.02	34.23
RU	2	10.27	55.68	10.26	34.06
	4	11.06	65.26	6.99	27.75
RC	0.25	12.96	59.52	8.09	32.40
	0.50	10.29	61.41	8.23	30.36
	0.75	8.76	60.51	9.55	29.95
D–Incr	5%	7.66	61.13	9.14	29.73
	10%	21.19	64.70	7.67	27.63
	15%	5.73	55.02	9.28	35.69
	20%	7.92	61.11	8.43	30.46
%Neg	0%	0.80	58.59	13.28	28.13
	20%	1.18	66.41	9.47	24.12
	40%	2.01	56.87	6.92	36.21
	60%	37.07	60.11	5.85	34.04
	80%	22.22	61.33	3.35	35.31
	100%	1.15	59.54	12.80	27.65
Overall		10.67	60.48	8.62	30.90

Table 2.6: Comparative computational results part 2 (5,000 schedules).

As already stated the algorithm of Gu et al. (2013) compares with the results of Vanhoucke (2010) based on a 5 minutes stopping criterion instead of the 5,000 schedules criterion. This means that in order to properly compare our own method with the former we should also terminate after 5 minutes. We have, however, chosen to employ a stopping

criterion of 12,500 schedules with a maximum time limit of 5 minutes per instance. We furthermore use a 2.5 GHz Dual Core processor, whereas Gu et al. (2013) use a computing cluster where each node consists of two 2.8 GHz 6–Core processors.

The results of the comparison with the algorithm of Gu et al. (2013) can be found in table 2.7. The results used of the latter are those of the CP–LR since the authors show this method clearly performs best compared to others. This CP–LR is one of several alternatives tested by them and constitutes a hybrid approach of constraint programming and Lagrangian relaxation.

Table 2.7 is constructed similarly to table 2.5, of which the latter compares our GA2 with the scatter search of Vanhoucke (2010). This way, a similar analysis can be made and the following conclusions can be drawn:

- The overall results show a better performance of our GA2 than the CP–LR of Gu et al. (2013).
- In particular our *%Best* is larger in nearly all cases.
- Comparatively our method performs better as *Act* increases meaning that it is more robust as the project size increases. This can be seen by an increasing *%Best* of our method as the number of activities increases, whereas the results of Gu et al. (2013) show alternating decreases and increases. The gap between both methods in terms of *%AvDev* furthermore becomes larger in favour of GA2 with an increasing number of activities.

In table 2.8 we show the average computation times of our own algorithm (*AvTime*) and the percentage of instances for which the 5 minutes limit was actually reached (*%Limit*). Only for a very small number of instances is the 5 minutes limit actually reached.

Although the improvement of our method in table 2.7 is smaller than the one in tables 2.5 and 2.6, it is important to take into account that our algorithm on average only takes 10.28 seconds, see table 2.8, whereas Gu et al. (2013) always use 5 minutes. We furthermore employ a computer with a slower processor to test our method. As such, based on both tables 2.7 and 2.8 it can be concluded that our proposed scheduling method and GA2 are not only faster but also provide better results than the methodology of Gu et al. (2013). In particular, the number of generated schedules is set to 12,500 since this can be seen as the cut–off point in terms of number of schedules where our GA2 outperforms the CP–LR of Gu et al. (2013).

		This chapter			Gu et al. (2013)		
		%AvDev	*%Best*	*%Feas*	*%AvDev*	*%Best*	*%Feas*
Act	25	194.05	67.13	99.58	190.05	85.83	100.00
	50	289.57	79.53	99.54	285.36	54.28	100.00
	75	149.66	89.46	99.07	153.51	70.69	99.98
	100	201.62	89.81	98.77	208.08	61.13	99.68
OS	0.25	161.83	84.35	98.75	161.09	76.40	99.91
	0.50	203.33	81.43	99.11	207.53	68.26	99.83
	0.75	260.60	76.08	99.86	257.26	59.31	100.00
RU	2	248.57	78.34	99.06	248.18	73.44	99.83
	4	169.16	82.86	99.42	169.18	62.55	100.00
RC	0.25	333.43	79.35	99.03	65.50	82.83	100.00
	0.50	139.87	81.02	99.15	338.33	62.09	99.74
	0.75	153.61	81.44	99.55	224.44	59.06	100.00
D–Incr	5%	228.47	82.49	97.04	222.64	74.16	99.65
	10%	138.93	83.74	99.93	142.94	69.68	100.00
	15%	328.22	77.27	100.00	323.67	65.32	100.00
	20%	140.15	78.98	100.00	146.57	62.78	100.00
%Neg	0%	30.55	84.36	99.48	30.64	74.31	99.90
	20%	28.13	87.26	99.51	28.42	73.20	99.90
	40%	63.69	80.68	99.41	67.03	64.60	99.90
	60%	476.10	75.93	98.68	460.68	66.95	99.93
	80%	422.20	70.95	99.20	422.21	65.39	99.93
	100%	234.78	84.38	99.17	247.07	63.52	99.93
Overall		208.71	71.97	99.24	209.33	67.99	99.91

Table 2.7: Comparative computational results 12,500 schedules.

Act	*AvTime* (s)	*%Limit*
25	1.48	0.00
50	5.05	0.00
75	11.58	0.00
100	23.42	0.47
Overall	10.28	0.12

Table 2.8: Computation times 12,500 schedules.

2.7 Conclusions

In this chapter, we discussed the RCPSPDC with payments at activities' completion times. We proposed a new scheduling technique which makes use of rules to move activities. These rules focus on maximizing project NPV by delaying sets of activities with a negative cumulative NPV or advancing those with a positive NPV. An important part of our methodology is the construction of sets of activities which have to be moved together and can be built in two ways. A first method evaluates the predecessors and successors of

an activity and determines whether they should be included in the set or not. A second method focusses on the neighboring activities in the project schedule which means the activities under consideration need not be precedence related. A penalty function was also included for the schedules infeasible with respect to the project deadline. Each step of our algorithm has been extensively tested and two variants of a genetic algorithm were proposed. Our final procedure was compared with the best known results from literature and was shown to perform considerably better.

3

Payment models and net present value optimization for resource–constrained project scheduling

This chapter focuses on the single– and multi–mode resource–constrained project scheduling problem with discounted cash flows (RCPSPDC and MRCPSPDC) and three payment models. The contribution of the chapter is twofold. First, we extend a new scheduling technique, which moves activities in order to improve the project net present value. This more general version is applicable to multiple problem formulations and provides an overarching framework in which these models can be implemented. The changes in activity finish times take other activities and the possible changes in the finish times of these other activities into account, by forming a set of activities which is subsequently moved in time. The scheduling technique is implemented within a genetic algorithm metaheuristic and employs two penalty functions, one for deadline feasibility and one for non–renewable resource feasibility. Second, we test the proposed approach on several datasets from literature and illustrate the added value of each part of the algorithm. The influence of data parameters on the project net present value is highlighted. The detailed results provided in this chapter can be used as future benchmarks for each of the six models discussed.

3.1 Introduction

In this chapter, we focus on the maximization of the project NPV and discuss the RCPSP with discounted cash flows (RCPSPDC) and its multi–mode variant the multi–mode resource–constrained project scheduling problem with discounted cash flows (MRCPSPDC). Furthermore, we apply three payment models to these two problem formulations. The three payment models discussed are payments at activities' completion times (PAC), progress payments (PP) and payments at event occurrences (PEO). These payment models determine the timing and amounts of cash inflows received and are based on different assumptions. Cash outflows are assumed to occur upon activity finish time for all models. The PAC model assumes cash inflows are received upon activity completion. This in turn implies a net cash flow can be calculated for each activity. In the PP and PEO models however, cash inflows occur at regular or irregular times throughout the project duration and are based on the project progress up until the payment time. From a solution methodology perspective, the added value of this chapter is a new **scheduling technique**, which can handle the PP and PEO models, on top of the PAC model.

	Timing	**Size**	**Timing & size**
Cash in	Payments at activities' completion times (PAC)	Progress payments (PP)	Progress based payment pattern (PBPP)
		Payments at event occurences (PEO)	Expense based payment pattern (PBPP)
Cash out	Payments at activities' completion times (PAC)	Resource usage costs	General capital model

Figure 3.1: Overview of the research on project scheduling with NPV optimization in chapter 3.

The problems discussed are relevant from a practical point of view since several possibilities exist for the receipt of cash flows during the project runtime. The manner in which these cash flows are received is however often beyond the control of the party responsible for executing the project. This highlights the need to analyze the effect of different payment models on the project schedule and its resulting NPV. Furthermore, individual activities may be executable in different modes, i.e. with different combinations of activity duration and resource demand. This way, additional flexibilities exist for the project schedule, which however also increase the problem complexity (Kolisch and Drexl, 1997) and as such require more complex algorithms to properly solve the problem.

The remainder of this chapter is organized as follows. In section 3.2 we give an overview

of the existing literature and section 3.3 discusses the single– and multi–mode RCPSPDC along with the investigated payment models. In section 3.4 we go into detail about our proposed scheduling approach, as part of the metaheuristic presented in section 3.5. The results of our computational experiments are discussed in section 3.6. Finally, in section 3.7 we formulate our conclusions.

3.2 Literature overview

In this section, we provide a literature overview of the problems under consideration. We however only include research done after the general literature overview on NPV optimization of Herroelen et al. (1997). The distinction is made between the RCPSPDC (single–mode) and the MRCPSPDC (multi–mode).

3.2.1 Single–mode

A recent overview of the RCPSPDC with the PAC model is given in chapter 2 (Leyman and Vanhoucke, 2015). To the best of our knowledge only three papers exist which discuss other payment models for the RCPSPDC. The first is the paper of Sepil and Ortac (1997), in which the authors apply the PP model to the RCPSPDC and propose three different heuristic rules. These rules are applied in a single-pass greedy forward algorithm and determine the priority given to the different feasible activities at a specific time instance. The first heuristic gives priority to the activities with the highest NPV, whereas the second one applies a pairwise comparison of the NPV of all feasible activities. Finally, the third priority rule takes the slope of the activity profit curves into account. The other two papers of Möhring et al. (2001) and Möhring et al. (2003) tackle project scheduling problems with irregular objective functions and propose a uniform methodology for solving resource–constrained project scheduling problems based on minimum cuts. The focus of both papers lies on the mathematical problem formulation and a Lagrangian relaxation based approach to solve the problems to optimality. The authors conclude that the relaxed problem can be solved efficiently by minimum cut computations.

3.2.2 Multi–mode

Table 3.1 provides details of the research done for the MRCPSPDC since the literature overview of Herroelen et al. (1997).

- The objective of each paper can be found in the second (NPV) and third (Dur) column. Four papers combine NPV maximization and makespan minimization in a single objective. Mika et al. (2005) only work with positive cash flows for the

NPV objective (footnote [1] in table), whereas Kazemi and Tavakkoli-Moghaddam (2010) include a robustness measure as part of their makespan minimization objective (footnote [2] in table).

- Columns four to seven display the payment models used in the different research papers. These models constitute the PAC, PP, PEO and lump sum payment (LSP) variants.
- In columns eight to ten the required resource types are displayed. These resources include renewable resources (RR), non–renewable resources (NRR) and capital (C). If a cost is assigned to these resource types and these costs are included in the NPV objective, a footnote [3] is included.
- The final two columns show whether the authors additionally discuss a client–contractor trade–off (CC) and whether a bonus/penalty (B/P) structure is included with respect to the project deadline.

Based on table 3.1 it can be concluded that a multitude of problem formulations exist in literature when discussing NPV optimization in a multi–mode context.

	Objective		Payment models				Resources			Other	
Authors	**NPV**	**Dur**	**PAC**	**PP**	**PEO**	**LSP**	**RR**	**NRR**	**C**	**CC**	**B/P**
Özdamar and Dündar (1997)	X				X				X		X
Özdamar (1998)	X				X				X		X
Ulusoy and Cebelli (2000)	X				X		X	X		X	
Ulusoy et al. (2001)	X			X	X	X	X	X			
Mika et al. (2005)	X[1]		X	X	X	X	X	X			
Chen and Chyu (2008)	X		X				X[3]	X[3]			
Seifi and Tavakkoli-Moghaddam (2008)	X	X		X	X	X	X	X			
Kavlak et al. (2009)	X		X	X			X	X		X	
Chen et al. (2010)	X				X		X[3]	X[3]			X
Kazemi and Tavakkoli-Moghaddam (2010)	X	X[2]			X	X	X				
Azimi et al. (2011)	X	X	X				X				
Aboutalebi et al. (2012)	X	X			X		X[3]				
Chen and Zhang (2012)	X				X		X[3]	X[3]			X
Hosseini et al. (2014)	X					X	X	X			

[1]: Only positive cash flows are used, [2]: Robustness is included in the duration measure, [3]: A cost is assigned to this resource type.

Table 3.1: Literature overview MRCPSPDC.

3.3 Problem description

In this section, we first discuss the mathematical models for both the single–mode and multi–mode RCPSPDC. We present the PAC model since this payment model is most commonly used, especially in the single–mode literature (Herroelen et al., 1997; Leyman and Vanhoucke, 2015; Vanhoucke et al., 2001b). Both mathematical models are subsequently extended to the PP and PEO payment models.

3.3.1 Payments at activities' completion times

We use the activity–on–the–node (AoN) representation for a network $G(N, A)$ with N the set of project activities or network nodes and A the set of precedence relations or network arcs. The activities are numbered from the start dummy 0 to the end dummy $n + 1$. Each activity i ($i \in N = \{1, \dots n\}$) has a duration d_i, a cash in– and ouflow, respectively $c_{i,in}$ (> 0) and $c_{i,out}$ (< 0), and a RR demand r_{ik}^{ρ} of type k. Each RR of type k ($k \in R^{\rho} = \{1, \dots, |R^{\rho}|\}$) has a constant availability of a_k^{ρ} throughout the project duration. A time–lag of zero is assumed for the precedence relations, and the project has a deadline δ_{n+1}. The finish time of each activity i is contained in the decision variables f_i.

Conceptually, the RCPSPDC with PAC can be formulated as follows:

$$\text{Maximize} \sum_{i=1}^{n} (c_{i,in} + c_{i,out}) \cdot e^{-\alpha f_i} \tag{3.1}$$

Subject to:

$$f_i \leq f_j - d_j, \quad \forall (i,j) \in A \tag{3.2}$$

$$\sum_{i \in S(t)} r_{ik}^{\rho} \leq a_k^{\rho}, \quad \forall k \in R^{\rho}, \ t = 1, \dots, \delta_{n+1} \tag{3.3}$$

$$f_{n+1} \leq \delta_{n+1} \tag{3.4}$$

$$f_i \in int^{+} \quad \forall i \in N \tag{3.5}$$

The objective function (3.1) maximizes the project NPV by discounting the cash in– and outflows to each activity's finish time. Hence the objective function can be simplified to $\sum_{i=1}^{n} c_{i,net} \cdot e^{-\alpha f_i}$, with $c_{i,net} = c_{i,in} + c_{i,out}$. Constraints (3.2) enforce the precedence constraints, whereas constraints (3.3) impose the renewable resource limits, with $S(t)$ the set of activities in progress at time t ($S(t) = \{i \in N : f_i - d_i \geq t \wedge f_i < t\}$). Constraint (3.4) makes sure the deadline is met, and finally constraints (3.5) state that the decision variables should be integers.

The RCPSPDC model from (3.1)–(3.5) can be extended to its multi–mode variant, based on the model of Van Peteghem and Vanhoucke (2014) for the multi–mode RCPSP

(MRCPSP). Each activity now has a duration d_{im_i}, which differs depending on the mode m_i selected for each activity i out of a set $|M_i|$ of different modes with $M_i = \{1, \ldots, |M_i|\}$. Furthermore, each mode has a unique RR demand $r^{\rho}_{im_ik}$ per resource type k and a NRR demand $r^{\nu}_{im_il}$ per resource type l. Each (N)RR of type k (l) has an availability of a^{ρ}_k (a^{ν}_l), with $k \in R^{\rho} = \{1, \ldots, |R^{\rho}|\}$ ($l \in R^{\nu} = \{1, \ldots, |R^{\nu}|\}$). Constraints (3.2) are adjusted to (3.6) and (3.3) to (3.7) to include the different modes. Additionally constraints (3.8) ensure that the total NRR availability is not exceeded whereas constraints (3.9) make sure that each activity's mode is selected from the set of available modes for that activity.

$$f_i \leq f_j - d_{jm_j}, \quad \forall (i,j) \in A \tag{3.6}$$

$$\sum_{i \in S(t)} r^{\rho}_{im_ik} \leq a^{\rho}_k, \quad \forall k \in R^{\rho}, \ \forall m_i \in M_i, \ t = 1 \ldots, \delta_{n+1} \tag{3.7}$$

$$\sum_{i}^{n} r^{\nu}_{im_il} \leq a^{\nu}_l, \quad \forall l \in R^{\nu}, \ \forall m_i \in M_i \tag{3.8}$$

$$m_i \in M_i, \quad \forall i \in N \tag{3.9}$$

3.3.2 Progress payments

In the PP model, payments are made at regular time intervals, and the final payment is made at project completion. For a project with a deadline of 19, this could mean that payments are made e.g. every 5 time instances, namely at time instances 5, 10, 15 and 19. In terms of changes in the model's objective function (3.1), cash inflows p_h occur at $H-1$ regular payment times every T time instances. The final payment p_H is incurred at project completion. Each of these payments amount to the monetary value of the work done since the previous payment time. The cash inflow of each activity is furthermore calculated based on a profit margin applied to each activity's cash outflow. Assume for instance an activity with a cash outflow of -100 and a profit margin of 20%. In this case the activity's cash inflow is 120 (= 100 * 1.20). The objective function can be adjusted in the following way:

$$\text{Maximize} \sum_{h=1}^{H-1} p_h \cdot e^{-\alpha T \cdot h} + p_H \cdot e^{-\alpha \delta_{n+1}} + \sum_{i=1}^{n} c_{i,out} \cdot e^{-\alpha f_i} \tag{3.10}$$

It is important to stress that Ulusoy et al. (2001), Mika et al. (2005) and Seifi and Tavakkoli-Moghaddam (2008) discuss an alternative to PP, namely the equal time intervals (ETI) model. The major difference between these two models is that for PP the interval between payments is given, whereas for ETI the number of payments is used to determine

the payment times. Recall the earlier example for PP with payments at times 5, 10, 15 and 19. Assume we state a total of 4 payments, then the payment scheme for ETI would be the same as for PP, since ETI assumes the final payment interval to be smaller than or equal to the other intervals. Both the PP and ETI model also involve payments which occur with a fixed payment interval, with the final payment possibly occurring earlier. Since we do not aim to determine either the payment interval size or the number of payments we treat both models as the PP model.

The major difference between the PP and PAC model can be seen as follows. In the PAC model the payment times are not set in advance since cash inflows occur at activity completion times, which depend on the actual schedule. The payment amounts however are known in advance since they are equal to the cash inflows of the individual activities. For the PP model on the contrary the payment times are determined in advance whereas the payment amounts depend on the actual schedule, i.e. the work completed since the previous payment.

3.3.3 Payments at event occurrences

The payments at event occurrences (PEO) model can be seen as an irregular variant of the PP model. Whereas in the latter case cash inflows are received every T time instances, except for the final payment which may be irregular, in the PEO case all payment are of the irregular type. Given a project deadline of 19, payments for the PEO model could occur at time instances 4, 10, 16 and 19 for example. The objective function can be altered in the following way, with T_h the time instance at which payment h is received:

$$\text{Maximize} \sum_{h=1}^{H} p_h \cdot e^{-\alpha T_h} + \sum_{i=1}^{n} c_{i,out} \cdot e^{-\alpha f_i} \tag{3.11}$$

3.3.4 Problem complexity & classification

Blazewicz et al. (1983) have proven the optimization variant of the RCPSP, and by extension the RCPSPDC, to be strongly NP–hard. It can be represented as $m, 1|cpm, \delta_n, c_i|npv$ for the PAC model and as $m, 1|cpm, \delta_n, per|npv$ for the PP and PEO models according to the classification scheme of Herroelen et al. (1999), and as $PS|prec|\sum C_i^F \beta^{C_i}$ according to Brucker et al. (1999).

The optimization variant of the MRCPSP is also strongly NP–hard, whereas the decision variant has been proven to be NP–complete by Kolisch and Drexl (1997), once at least two non–renewable resources are included. This implies that the MRCPSPDC is at least as complex. The MRCPSPDC can furthermore be formulated as $m, 1T|cpm, \delta_n, disc, mu, c_i$

$|npv$ (PAC) or $m, 1T|cpm, \delta_n, disc, mu, per|npv$ (PP and PEO) following the scheme of Herroelen et al. (1999) and as $MPS|prec|\sum C_i^F \beta^{C_i}$ following Brucker et al. (1999).

3.4 Schedule generation

In this section, we go into detail about our schedule generation scheme and its applications to the payment models of section 3.3. We start with the initial schedule generation and penalty functions for both the single– and multi–mode cases. We discuss the details of our activity move rules as crucial improvements of the initial schedule. An overview of the entire schedule generation is given in figure 3.2. We assume that each schedule is constructed based on an ordering of activities provided by our metaheuristic (section 3.5), namely a priority list (PL). The list holds the order in which the activities should be scheduled and is precedence feasible. If the problem discussed involves the trade–off between different activity modes, a mode list (ML) is also used. The ML contains the selected mode for each activity.

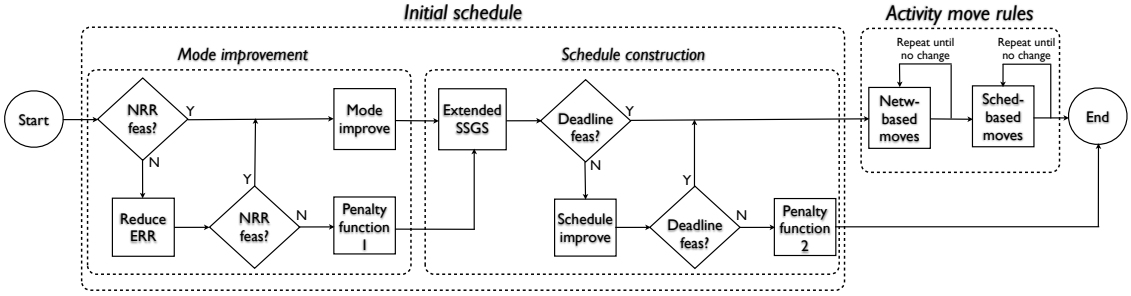


Figure 3.2: Schedule generation flow.

3.4.1 Initial schedule

We first discuss the mode improvement steps, which are only applicable for the MR-CPSPDC, and continue with the schedule construction which is used for both the single– and multi–mode case.

3.4.1.1 Mode improvement

In this section, we discuss three steps which make changes on the ML to improve the NRR feasibility and to help achieve a shorter project duration to ensure the deadline feasibility of the resulting schedule.

ERR reduction: The first step involves the calculation and feasibility test of the NRR. We calculate the solution's Excess of Resource Request (ERR) (Van Peteghem and Vanhoucke, 2010) defined as $ERR = \sum_{l=1}^{|R^\nu|}(\max(0, \sum_{i=1}^{n} r_{iml}^{v} - a_l^\nu))$. This value is 0 if the demand of each NRR does not exceed its availability and is larger than 0 if any of these constraints are violated. If $ERR > 0$ we apply the feasibility improvement method of Van Peteghem and Vanhoucke (2011), which decreases the ERR by changing modes of activities. The method is repeated until either no further improvements can be made or until the ERR has been reduced to 0. This step corresponds with the first *"NRR feas?"* and *"Reduce ERR"* in figure 3.2.

ERR penalty function: If the ERR is still positive we apply a penalty function. In the MRCPSP literature several penalty functions exist for makespan minimization, but when the goal is to maximize project NPV only the function of Mika et al. (2005) for positive cash flows exists. Since we consider both positive and negative cash flows, we propose the following alternative, with $NPV_{ERR-Infeas}$ the NPV of a solution with $ERR > 0$:

$$\begin{cases} NPV = -Y_1 + NPV_{ERR-Infeas} \cdot Y_2^{ERR} & \text{if } NPV_{ERR-Infeas} \geq 0 \\ NPV = -Y_1 + \frac{NPV_{ERR-Infeas}}{Y_2^{ERR}} & \text{otherwise} \end{cases} \tag{3.12}$$

- Y_1 constitutes a large fixed value to be subtracted from the project NPV to ensure that the infeasible solution's NPV is considerably worse than that of any other feasible solution in the solution set.
- Y_2 has a fractional value between 0 and 1 to reduce the project NPV based on the ERR. The larger the ERR the more the project NPV is reduced.

This step corresponds with the second *"NRR feas?"* and *"Penalty function 1"* in figure 3.2.

Project duration improvement: If the ERR is equal to 0, we calculate the critical sequence lower bound (CSLB) of Stinson et al. (1978). If this bound is larger than the best deadline–feasible project duration found so far, two of the mode improvement methods of Van Peteghem and Vanhoucke (2011) can be applied. The first method is the critical path (CP) improvement which aims to minimize the CP length. The second method focuses on work content (WC) improvement to reduce the total work content of the proposed ML. This step corresponds with *"Mode improve"* in figure 3.2.

3.4.1.2 Schedule construction

For both the RCPSPDC and the MRCPSPDC, the next step in our algorithm is the construction of an initial schedule with the serial schedule generation scheme (SSGS) of Kelley (1963). The initial schedule is then extended by applying an adjusted version of the improvement method of Van Peteghem and Vanhoucke (2010), which aims to reduce each activity's finish time by applying mode changes. For each activity, starting from the first activity in the PL, this method randomly selects a mode different from the current one. If the mode change does not increase the ERR, it aims to reschedule the activity with an earlier finish time. If a finish time reduction can be achieved, we retain the mode change and continue with the next mode of the current activity. Once all modes of an activity have been considered, we move on to the next activity in the PL until we reach the end of the PL. Finally, for each activity the method is applied with a fixed probability MI (mode improvement).

If the schedule is deadline–feasible we continue with the activity move rules of section 3.4.2. If the schedule turns out to be infeasible with respect to the project deadline (*D–Infeas*), we apply one iteration of the forward–backward improvement method of Li and Willis (1992) (*"Schedule improve"* in figure 3.2), which reduces the project duration as much as possible. If the schedule is then deadline–feasible we continue with the activity move to improve the project's NPV. Otherwise, we use the penalty function of chapter 2 (Leyman and Vanhoucke, 2015):

$$\begin{cases} NPV = -Y_3 + NPV_{D-Infeas} \cdot Y_4^{f_{n+1}-\delta_{n+1}} & \text{if } NPV_{D-Infeas} \geq 0 \\ NPV = -Y_3 + \frac{NPV_{D-Infeas}}{Y_4^{f_{n+1}-\delta_{n+1}}} & \text{otherwise} \end{cases} \tag{3.13}$$

- Y_3 is similar to Y_1 in function (3.12) and is a large fixed value.
- Y_4 is a fractional value similar to Y_2 of function (3.12).

If a solution has both an $ERR > 0$ and a project duration larger than the deadline, we first apply function (3.12) and subsequently function (3.13). The values of Y_1, Y_2, Y_3 and Y_4 are tested in section 3.6.2.

3.4.2 Activity move rules

In this subsection, we give an overview of the rules for delaying activities which start from the initial forward schedule of the SSGS. We go into detail about NPV–profiles and then discuss the generalized rules applicable to all three payment models. The rules correspond with the *"Activity move rules"* in figure 3.2.

To properly illustrate these improvements we use the example shown in table 3.2 and in figure 3.3 and apply the PP model. The left side of the figure displays the project network with the cash outflows for each activity. A profit margin of 20% is used to calculate the activity cash inflows. We assume that the project has both a single RR and NRR with availabilities of respectively 5 and 21. Finally, the project deadline is 18. Note that the maximum NRR demand is 24, larger than the available 21, meaning that the NRR is not redundant (Sprecher et al., 1997).

Act	**Mode**	**Dur**	**RR**	**NRR**
1	1	3	3	3
	2	4	2	1
2	1	1	4	3
	2	5	5	1
3	1	4	3	2
4	1	5	2	3
5	1	2	4	2
	2	1	5	5
6	1	2	2	4
	2	3	3	3
7	1	1	2	2
8	1	2	4	1
	2	1	5	2

Table 3.2: Data example.

The initial RR, NRR and deadline feasible schedule of the example with PL (1, 2, 4, 3, 5, 7, 6, 8) and ML (1, 1, 1, 1, 1, 1, 1, 1) can be found on the right in figure 3.3. As an example, the set $S(t)$ of activities in progress at time t consists of activities 3 and 4 at time 6. The PL constitutes the order in which the scheduler considers the activities, whereas the ML shows the selected modes for each activity. Both lists are part of our schedule representation and are discussed in more detail in section 3.5.1. The horizontal axis of figure 3.3 shows the time and the vertical one the RR usage. Since the activity move rules do not make any changes on the ML, the ERR of the example remains equal to zero throughout the example schedules in this section. We assume payments occur every 5 time units for the PP model, meaning payments happen at times 5, 10, 15 and 18. We use the same discount rate of 0.167% as employed by Vanhoucke et al. (2003) for the example. The example's initial NPV with this discount rate is 42.73 (= 7.80 + 3.93 + 1.96 + 5.91 + 11.61 + 1.92 + 5.70 + 3.90).

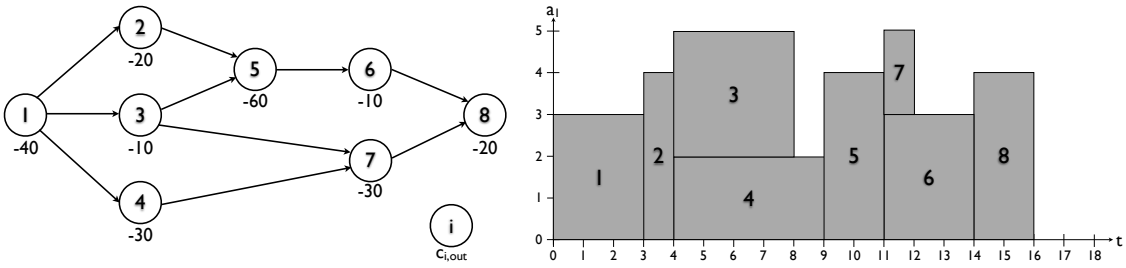


Figure 3.3: Network & initial schedule example.

3.4.2.1 NPV–profiles

For the PP and PEO models the NPV profiles or activity profit curves have a typical sawtooth pattern, as discussed in Kazaz and Sepil (1996) and Vanhoucke et al. (2003). For the discount rate we follow the reasoning, presented by Kazaz and Sepil (1996) and used by Vanhoucke et al. (2003), that for PP and PEO the NPV curves of an activity can be approximated by a piecewise linear one, if the daily discount rate is smaller than 2%. Kazaz and Sepil (1996) furthermore show that such a discount rate amounts to an annual rate of 730%, which is deemed very unrealistic.

Two examples of such a sawtooth pattern are shown in figure 3.4. The possible finish times of the activity can be found on the horizontal axis, ranging from the critical path method's (CPM) earliest finish time (ef_i) to the latest finish time (lf_i, taking δ_{n+1} into account). The activity NPV is shown on the vertical axis. T_i is used to signify the i^{th} payment time in the example for both payment models, in line with the notation used in section 3.3. The graph on the left is typical for the PP model, where payments occur at regular time intervals, e.g. every 5 time units, and constitutes the activity profit curve of activity 2 in the example. In terms of an activity's NPV profile this means that a later peak in the NPV is always lower than a previous one. For the PEO model, we employ the same example data, but assume payments occur at times 9, 11 and 18. The right graph in figure 3.4 corresponds with activity 4 in the example. As can be observed, for the PEO model a later peak may be higher than an earlier one.

In order to determine how much an activity or set of activities should be delayed we need the finish times at which each activity reaches a peak in terms of NPV for both the PP and PEO models. Since the cash inflow for an activity is received proportionally to the percentage of the activity completed at the payment time, we can conclude that peaks are always reached if an activity's finish time equals a payment time (Vanhoucke et al., 2003). In such a case, both the cash in–and outflows occur at the same time and the difference between the payment and receipt of cash is minimized. As an example, consider the left

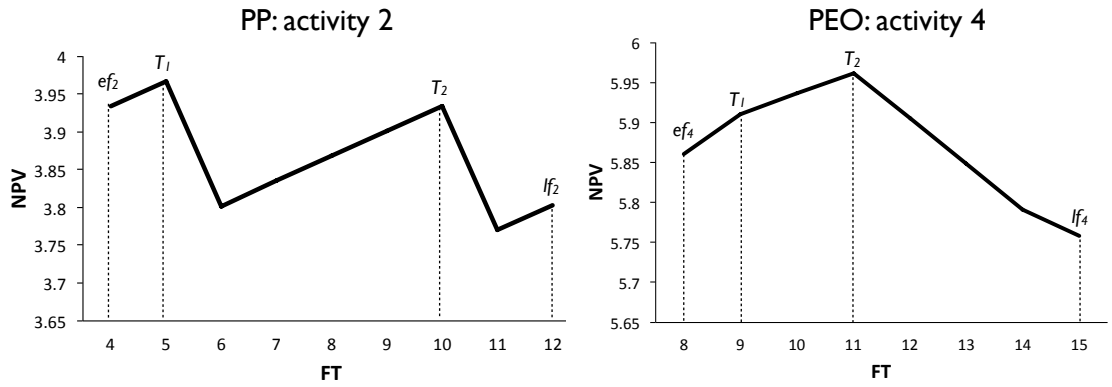


Figure 3.4: Activity profit curve PP and PEO.

graph of figure 3.4, which illustrates that the peaks occur at payment times 5 and 10, and that the NPV at time 10 is lower than the NPV at time 5. For the PEO model peaks occur at payment times 9 and 11 (right graph of figure 3.4). Since the NPV at time 11 is however higher than the NPV at time 9, we only take the peak at time 11 into account.

For the PAC model on the contrary each activity either has a positive or negative cash flow, which means that the activity has a strictly decreasing or increasing NPV–curve, and that it should be scheduled as soon as possible or as late as possible respectively. As a result, the solution approach of chapter 2 for the PAC model (Leyman and Vanhoucke, 2015) cannot be used for the PP and PEO models, since it only considers strictly increasing or decreasing linear profit curves. In sections 3.4.2.2 and 3.4.2.3 more complex activity move rules are presented which take piecewise linear activity profit curves (such as for the PP and PEO models) into account. These rules can be applied to all three payment models as part of a single approach.

3.4.2.2 Network–based moves

The first set of activity move rules or network–based moves delays sets of activities based on the project network's precedence relations. Starting from the schedule generated by the forward SSGS we delay individual activities or sets of activities to improve the project NPV. We start from the last activity in the PL and determine for each activity whether it is at a peak in the profit curve or not. If the activity under consideration (called the current activity hereafter) does not have a finish time which also constitutes a peak in the activity profit curve and at least one later peak exists, two cases are possible:

Single activity: If a delay is possible based on the activity's successors we set the activity's finish time equal to the next peak and check the RR availability. If sufficient

resources are available we retain the new activity finish time. Otherwise we reduce the new finish time by 1 and repeat the RR check. The reduction by 1 of the new finish time is only applied while both a delay is possible and such a delay leads to an increase in the project NPV. Once either condition is violated, the search for a new activity finish time terminates.

Algorithm 3 Get all successors

GetAllSuc (current activity i, start activity k, $setAct[]$, $NPV_{cum,0}$)
  $\forall j \in S_i$
    If $j \notin setAct \wedge f_j - d_{j(m_j)} = f_i$
      Add j to $setAct$
      Add NPV_j to $NPV_{cum,0}$
      GetAllSuc (j, k, $setAct$, $NPV_{cum,0}$)
    End if
  $\forall j \in P_i$
    If $j \notin setAct \wedge f_j + d_{i(m_i)} = f_i \wedge j \neq k \wedge f_j \geq f_k \wedge f_j < lf_j \wedge \exists np_j$
      Add j to $setAct$
      Add NPV_j to $NPV_{cum,0}$
      GetAllSuc (j, k, $setAct$, $NPV_{cum,0}$)
    End if
  Return $setAct$ and $NPV_{cum,0}$

Activity set: If a delay is impossible due to at least one successor with a start time equal to the current activity's finish time, we apply algorithm 3. This algorithm finds the set of activities which should be considered together for a potential delay. Starting from the current activity all successors are added which start immediately after the current activity, and the procedure is recursively applied again for each of these successors. For all these activities, predecessors with a finish time not smaller than the current activity's finish time and for which a later peak in the activity profit curve exists, are also added. If no later peak exists for these predecessors, they are not added because they should not be delayed together with the other activities in the set. Predecessors of the current activity are not taken into account since these activities are considered when they are reached in the PL.

Once algorithm 3 returns to the current activity, we calculate each activity's allowable delay, based both on a later peak and on any successors not in the set. The global minimum of these allowable delays is the maximum allowable delay Δ and is calculated as follows: $\Delta = min(np_j - f_j, min(f_k - d_{k(m_k)} - f_j | k \in S_j, k \notin setAct) | j \in setAct)$, with $setAct$ the set of activities returned by algorithm 3 and np_j the next peak of activity j. $d_{k(m_k)}$ is used to signify that in case the multi–mode problem is solved the activity duration depends on the mode selected for that activity, whereas this is not the case for a single–mode problem. If no later peak exists for activity j, np_j is set to infinity. Next, we calculate $NPV_{cum,\Delta}$ which is the cumulative NPV of all activities in the set if their finish time would be increased

by Δ, and compare with the cumulative NPV of all activities in the set without a delay $NPV_{cum,0}$.

1. $NPV_{cum,\Delta} > NPV_{cum,0}$: we increase the finish time of each activity in the set by Δ and check whether the resulting schedule is RR feasible. If the schedule is feasible the new finish times are retained. Otherwise, we decrease Δ by 1 and repeat until we either find a RR feasible Δ or until Δ reaches 0. We only repeat this RR feasibility check while $NPV_{cum,\Delta} > NPV_{cum,0}$, since a delay always has to lead to a NPV increase.

2. $NPV_{cum,\Delta} \leq NPV_{cum,0}$: we calculate $NPV_{cum,pot}$, which is the potential cumulative NPV calculated by delaying individual activities in the set by more than Δ.

   Starting from the activity in the set with the latest finish time, we check whether for that activity j the following applies:

   $$min(np_j - f_j, min(f_k - d_{k(m_k)} - f_j | k \in S_j, k \notin setAct)) \\ \leq min(f_k + \Delta_k - d_{k(m_k)} - f_j | k \in S_j, k \in setAct) \quad (3.14)$$

   The left hand side of this inequality calculates the possible delay of the activity j based on its next peak np_j and based on the finish time f_k of any succeeding activities not in $setAct$. The right hand side determines the activity's possible delay based on any delays Δ_k of successors k of j which are included in the set. Δ_j can furthermore be different for any activity j in the set. The inequality as a whole tests whether activity j can indeed be delayed further based on additional delays of any successors k in the set.

   Subsequently, we evaluate the following inequality:

   $$\Delta < min(np_j - f_j, min(f_k - d_{k(m_k)} - f_j | k \in S_j, k \notin setAct)) \quad (3.15)$$

   The right hand side of this inequality is the same as the left hand side of inequality (3.14), whereas the left hand side of (3.15) constitutes the original delay Δ calculated for the set as a whole. Inequality (3.15) is used to evaluate whether activity j can be delayed by more time units than Δ.

   If both inequalities (3.14) and (3.15) are met we set $\Delta_j = min(np_j - f_j, min(f_k + \Delta_k - d_{k(m_k)} - f_j | k \in S_j, k \notin setAct))$, which implies that j may now be delayed further because of the later finish time of its successors in the set, i.e. $\Delta_j > \Delta$. We also add activity j's NPV given Δ_j to $NPV_{cum,pot}$. If both inequalities are not satisfied, Δ is retained as Δ_j and we add the activity NPV of activity j given Δ to $NPV_{cum,pot}$.

We move on to the activity j in the set with the highest finish time which has not yet been considered, and repeat the same Δ_j calculations until all activities in $setAct$ have been considered. Once all activities in $setAct$ have been considered, we evaluate $NPV_{cum,pot}$.

(a) $NPV_{cum,pot} > NPV_{cum,0}$ and the delay is RR feasible for all activities in the set: we delay each activity j with Δ_j and the schedule is updated.

(b) $NPV_{cum,pot} > NPV_{cum,0}$ and the delay is not RR feasible for all activities in the set: The Δ_h for any activity h with an infeasible delay is decreased by 1. Furthermore, the values of each Δ_j corresponding to an activity j in the set with a finish time smaller than that of activity h are recalculated if a decrease in Δ_h has occurred. These Δ_j are recalculated because the decrease of Δ_h may lead to a violation of equation (3.14) due to a decrease in the right hand side. As such the possible delays of any such activity j need to be adjusted. We repeat the evaluation of $NPV_{cum,pot}$ for the entire set.

(c) $NPV_{cum,pot} \leq NPV_{cum,0}$: no NPV improvement can be achieved by the potential NPV calculations and the delay rule terminates for activity i of the PL.

If an activity on the contrary is scheduled at a peak, no later peaks exist, or the search for a delay is completed, we continue with the previous activity in the PL until the first activity in the PL is reached. Finally, if at least one change has occurred the procedure is repeated until no more delays are needed.

Figure 3.5 gives an overview of the flow of the activity move rules for an activity i. We assume algorithm 3 has been applied and we start by calculating the set's Δ and then evaluate Δ and the corresponding NPV change.

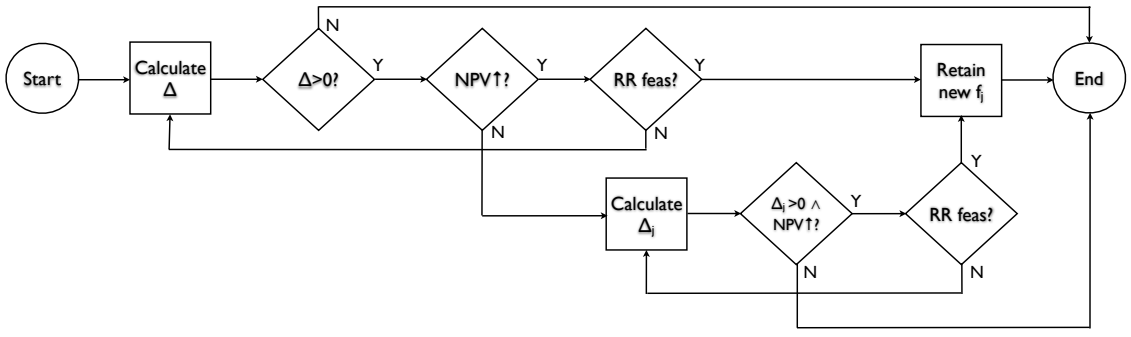


Figure 3.5: Flow of NPV improvement.

Finally, table 3.3 gives an overview of the notations used in this section. Additionally, the values of the notations are shown for two iterations of the example of section 3.4.2.4.

Notation	Definition	Example	
		Network–based	*Schedule–based*
i	Current act in PL	5	2
$setAct$	Set of act j returned by alg 3 or 4	{5, 6, 7, 8}	{2, 4}
Δ	Max allowable delay for $setAct$	2	1
Δ_j	Individual delay of act $j \in setAct$	2, 2, 2, 3	/
$NPV_{cum,0}$	NPV of $setAct$ without delay	23.18	10.08
$NPV_{cum,\Delta}$	NPV of $setAct$ with delay Δ	23.16	**10.10**
$NPV_{cum,pot}$	NPV of $setAct$ with Δ_j $(\forall j \in setAct)$	**23.19**	/

Table 3.3: Overview of notations activity move rules.

3.4.2.3 Schedule–based moves

The second set of activity move rules or schedule–based moves delays sets of activities based on their neighboring activities in the project schedule. These neighboring activities may, but need not, be precedence related. Instead, in this second set of rules we group activities together if one activity's finish time equals another's start time.

The network–based rules are extended because activities may not be delayed due to non–precedence related activities which cause a resource conflict. The only difference with the methodology discussed in section 3.4.2.2 is the way sets of activities are constructed, which means algorithm 3 needs to be adjusted. Algorithm 4 shows the adjusted version which adds neighboring activities to the set even if they are not precedence related.

Algorithm 4 Get all later neighbors

GetLaterNeighb (current activity i, start activity k, $setAct[]$, $NPV_{cum,0}$)
  $\forall j \in N \wedge f_i = f_j - d_{j(m_j)}$
    If $j \notin setAct$
      Add j to $setAct$
      Add NPV_j to $NPV_{cum,0}$
      GetLaterNeighb (j, k, $setAct$, $NPV_{cum,0}$)
    End if
  $\forall j \in N \wedge f_i - d_{i(m_i)} = f_j$
    If $j \notin setAct \wedge j \neq k \wedge f_j \geq f_k \wedge f_j < lf_j \wedge \exists np_j$
      Add j to $setAct$
      Add NPV_j to $NPV_{cum,0}$
      GetLaterNeighb (j, k, $setAct$, $NPV_{cum,0}$)
    End if
  Return $setAct$ and $NPV_{cum,0}$

3.4.2.4 Example

Let us illustrate both the network– and schedule–based rules on the example for the PP model.

1. **Network–based moves:** starting from the schedule in figure 3.3 and PL (1, 2, 4, 3, 5, 7, 6, 8), we first try to delay activity 8. This activity is however scheduled at a peak (time 15) and has no successors, so we move on to activity 6. This activity would have to be delayed together with its successor activity 8, but this would decrease the total NPV. Next, activity 7 can be delayed by 1 time unit to time instance 13, since from time 12 to 15 the slope of its activity profit curve is increasing. The delay is furthermore RR feasible. Next, we try to delay 5 but this is impossible due to activity 6. Hence, we apply algorithm 3 and the returned set is {5, 6, 7, 8}. Its minimal allowable delay is 2 because of the peaks of activities 6 and 7 at time 15. Doing so would however decrease the project NPV, so we calculate the potential NPV by delaying activity 8 with an additional time unit to time 18. Activity 8 can be delayed further because it has no successors and has a peak at time 18. As such, given the delay of 2 for the entire set it makes sense to additionally delay activity 8 by 1 time unit with a total delay of 3 time units. For activities 6 and 7 this implies that the right hand side of equation (3.14) is increased by 1 and that both activities could also be delayed by an additional time unit. Both activities however have a peak at time 15 (given a delay of 2 time units) and are not considered for further delay, although it is possible. Based on this additional delay of activity 8, the project NPV increases so we delay activities 5, 6 and 7 by 2 time units and 8 by 3 time units. The values for the notations of the delay of activity 5 and the resulting $setAct$ are shown as an example in the third column of table 3.3. The next activity in the PL is activity 3 which can be delayed by 2 time units because it has a peak at time 10. Then, activity 4 is not considered because its duration equals the payment interval size and hence its NPV curve only decreases. Activity 2 cannot be delayed due to activity 4, which is not a successor of activity 2. Finally, it is impossible to delay activity 1 due to its successor activity 2. Since we have reached the first activity in the PL and at least one change has occurred, the procedure is repeated. No further delays are possible however. The resulting schedule is shown at the left in figure 3.6 and the project NPV is 42.80.

2. **Schedule–based moves:** the first activity to be considered is activity 5, since activities 8, 6 and 7 are scheduled at a peak. Activity 5 however cannot be delayed to due activities 6 and 7. We apply algorithm 4 and its returns the set {5, 6, 7}. Delaying this set any further would however decrease the project NPV. Next, is activity 3 which is also scheduled at a peak, so we move on to activity 4. This activity's NPV however only decreases, so we skip it. Activity 2 can be delayed but this is prohibited by activity 4. After applying algorithm 4 we get the set {2, 4}

which can be delayed by 1 time unit. It is crucial to take into account that this delay would never have been possible based on the first set of rules. The values for the notations of the delay of activity 2 and the resulting *setAct* are shown as an example in the fourth column of table 3.3. Finally, we reach activity 1 which, due to the delay of activities 2 and 4, can also be delayed by 1 time unit because it reaches a peak at time 5. The method is repeated but no further delays are possible. The project NPV after both sets of rules have been applied is 42.89 and the project schedule can be seen at the right of figure 3.6.

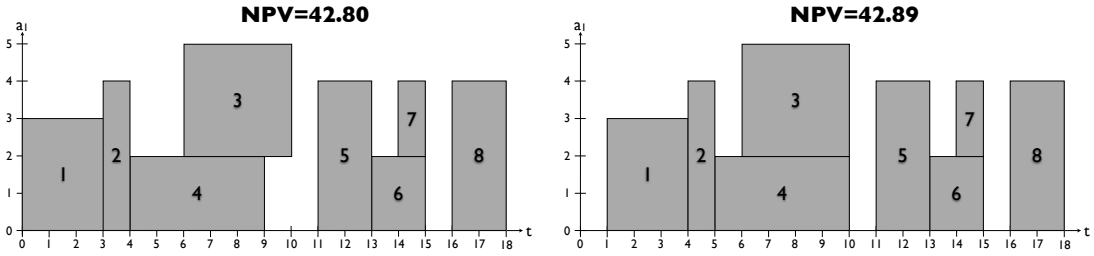


Figure 3.6: Schedule examples PP model.

3.5 Genetic algorithm

In this section, we discuss the details of our genetic algorithm (GA). Holland (1975) was the first to discuss the GA. The algorithm is based on evolutionary biology and makes use of selection, crossover and mutation operators to produce new solutions based on existing ones. The GA has already been extensively discussed in literature and the flow of the procedure can be described in the following way. A starting population with size $|P_0|$ is created either randomly or based on existing heuristics. The starting population P_0 is then reduced to a parent population P, which is used for the creation of the next generation of elements. From the parent population the selection operator chooses two parents. Each pair of parents is used by a crossover operator which combines characteristics of the parents to come up with new promising solutions. These new solutions or children are then mutated with a predefined percentage (M_1 for the PL and M_2 for the ML) and can be inserted into the parent population by a population update. This update retains a limited number $|R|$ of the best parents and replaces the rest by elements from the children set. An overview of the proposed GA, with the scheduler of section 3.4 as an integral part for the solution evaluation, is shown in figure 3.7. We continue this section by discussing each part of our GA in more detail.

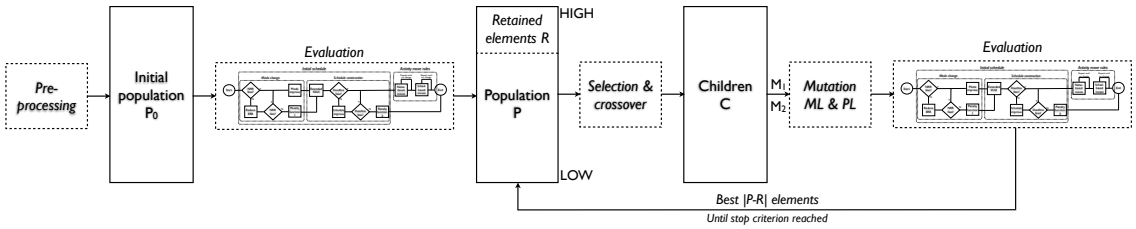


Figure 3.7: Genetic algorithm: procedure.

3.5.1 Representation

The PL has to be linked as closely as possible to the final schedule of the scheduling method, since in section 3.4 we move activities and change modes in the order of the PL. Hence we adjust the PL to the topological order (TO) representation discussed by Valls et al. (2004, 2003), and used for the RCPSP by Debels et al. (2006). The latter make sure the PL is precedence feasible and take the activity finish times into account when determining the order of activities in the list. We transform an element's PL into a TO representation once all steps of the scheduling procedure of section 3.4 have been applied. If two or more activities have the same finish time, ties are broken by randomly selecting which activity goes first. Recall in the example from section 3.4 that the initial PL was (1, 2, 4, 3, 5, 7, 6, 8). Based on the final schedule at the right of figure 3.6 the TO representation is (1, 2, 3, 4, 5, 6, 7, 8), with ties between activities 3 and 4, and between 6 and 7 broken randomly.

The ML is represented as is typically done in the MRCPSP literature (see e.g. Van Peteghem and Vanhoucke (2014)) and holds the selected mode per activity. Only feasible activity modes with respect to the preprocessing rules (section 3.5.2) are used.

3.5.2 Preprocessing

Before starting with the GA, several preprocessing steps are required. Specifically, the modes which are infeasible with respect to the RR, the inefficient modes and the redundant NRR need to be eliminated. The first need to be omitted because these modes will always violate the RR constraints, whereas the second are worse than other modes for the same activity due to e.g. requiring the same RR and NRR but having a longer duration. The omission of NRR is needed if for all activities the mode with the largest NRR is selected and the NRR availability is sufficient. For a more detailed description of these three preprocessing steps we refer to Sprecher et al. (1997).

3.5.3 Initial population

For the initial population P_0 we generate $|P_0|$ number of ML. We randomly select a mode for each activity which is not eliminated based on the preprocessing rules. In order to reduce this initial population with size $|P_0|$ of ML to a size of $|P|$, we apply one of two criteria (Van Peteghem and Vanhoucke, 2011) to evaluate each of the ML. The first is the sum of activity durations and the second is the total work content. The $|P|$ number of elements from P_0 with the lowest value for the selected evaluation criterion are retained for P whereas the rest is deleted. Additionally, for each of the retained ML we randomly generate a precedence feasible PL. These random solutions are then inserted in the GA's population P, and are evaluated based on the scheduler of section 3.4.

3.5.4 Selection

A selection operator is needed in a GA to ensure that the best elements are selected for reproduction in the crossover step. We test several alternatives, namely a roulette wheel selection, rank selection, four–tournament selection for both parents and the elite selection of chapter 2 (Leyman and Vanhoucke, 2015). The latter selects the father from the $|R|$ best elements in the population whereas the mother is chosen using a four–tournament selection.

3.5.5 Crossover

The crossover operator in a GA combines a number of existing parent elements and generates a number of children $|C|$ by recombining the information stored in the PL and ML. In this case, we always select two parents and produce two children. The alternatives tested are the one–point crossover, the two–point crossover, the MCUOX (Ulusoy et al., 2001) and the partially–mapped crossover.

3.5.6 Mutation

A mutation operator makes changes to the children created by the crossover with a certain probability. For the RCPSPDC we only apply mutation to the PL with a probability of M_1, whereas for the MRCPSPDC we also mutate the ML with probability M_2. We implement and test the mutation operators scramble, insertion, inversion and swap for both the PL and ML.

3.5.7 Evaluation and population update

Once the PL and ML of the children have been created, we apply the scheduling procedure discussed in section 3.4. We retain the best $|R|$ elements from the parent population, while the rest of the parents are replaced by the best children. $|R|$ also constitutes the size of the pool of elements from which the father is selected with the elite selection operator.

3.6 Computational results

This section shows the results of our computational experiments and goes into detail about the configuration of our algorithm. As shown in section 3.2 no papers exist in literature which discuss metaheuristics for the problems as defined in section 3.3. We however clearly illustrate the added value of each part of our proposed algorithm.

The 5,000 schedules stopping criterion as defined by Lova et al. (2009) is used for all tests. These authors define the number of generated schedules as the total number of times each project activity has received a new finish time divided by the total number of activities in the project. This definition implies that any change in activity finish or start time increases the number of generated schedules by $1/n$. Assume that the SSGS generates an initial schedule for a project with 5 activities, of which each activity has three possible modes. Additionally, a mode improvement method evaluates one other feasible mode for two of the activities, and afterwards 3 activities are delayed to improve the project NPV. The last two steps have no effect on the rest of the schedule. Based on the reasoning of Lova et al. (2009) the algorithm has generated 2 schedules ((= 5 + 2 x 1 + 3)/5). Finally, we employ a discount rate of 1% for all tests.

3.6.1 Test data

For the single–mode methods we use the project data of Vanhoucke (2010). For the multi-mode problems we use the MMLIB data of Van Peteghem and Vanhoucke (2014) since these authors show that the PSPLIB (Kolisch and Sprecher, 1996) and Boctor (Boctor, 1993) datasets contain shortcomings. The multi–mode data is extended with the cash flow data of Vanhoucke (2010), since no such data is included in the MMLIB datasets. An overview of the parameters of both relatively new datasets can be found in table 3.4.

No cash flow data and payment times data is however publicly available for the PP and PEO model. In terms of cash flow data, we use the 100% negative cash flow files of Vanhoucke (2010) as cash outflows and apply a profit margin ($\%Prof$) ranging from 0% to 100% in steps of 20%. For PP we assume payments occur every 10 time units, with

Parameter	Vanhoucke (2010)	MMLIB
Number of activities (Act)	25, 50, 75 or 100	50 or 100
Order strength (OS)	0.25, 0.50 or 0.75	0.25, 0.50 or 0.75
Renewable resource usage (RU)	2 or 4	/
Renewable resource factor (RF^{ρ})	/	0.50 or 1
Non–renewable resource factor (RF^{ν})	/	0.50 or 1
Renewable resource constraindness (RC)	0.25, 0.50 or 0.75	/
Renewable resource strength (RS^{ρ})	/	0.25, 0.50 or 0.75
Non–renewable resource strength (RS^{ν})	/	0.25, 0.50 or 0.75
Deadline increase (D–$Incr$)	5, 10, 15 or 20	5, 10, 15 or 20
Percentage negative cash flows ($\%Neg$) (PAC)	0, 20, 40, 60, 80 or 100	0, 20, 40, 60, 80 or 100
Profit margin ($\%Prof$) (PP & PEO)	0, 20, 40, 60, 80 or 100	0, 20, 40, 60, 80 or 100

Table 3.4: Parameter settings of test instances.

the final payment at project deadline, whereas for PEO we have generated payment times with a payment interval size randomly generated from [5;15].

With respect to *%Prof* used by the PP and PEO models the following has to be clarified. Since this profit margin has to be applied on each activity separately (Vanhoucke et al., 2003), this implies that the magnitude of the project NPV between the PAC model on the one hand and the PP/PEO models on the other hand will be different, with a smaller average NPV for the PAC model (tables 3.10 to 3.12) for the cash flows used in this research. The assumptions of the models (section 3.3) do not allow for the exact same cash flow data to be used for all three payment models since a positive profit margin for an activity (PP/PEO) always leads to a positive net activity cash flow, which would render the *%Neg* parameter for the PAC model useless. Vica versa the *%Neg* has no link to *%Prof*. The only way to link both parameters is by setting a profit margin on the project as a whole rather than on the activity level, but doing so would violate assumptions made in literature with respect to the PP model (Vanhoucke et al., 2003).

The problem set for the RCPSPDC contains 17,280 (= 720 x 4 x 6) problem instances and 103,680 (= (540 + 540 + 3,240) x 4 x 6) instances for the MRCPSPDC. All data of both Vanhoucke (2010) and Van Peteghem and Vanhoucke (2014), and the data generated for this chapter, along with the best known solutions are available online at *http://www.projectmanagement.ugent.be*.

3.6.2 Algorithm configuration

In this section, we configure our algorithm. We start by showing the values of the parameters for each payment model and the factors of the penalty functions. We furthermore give an overview of the added value of the mode improvement of section 3.4.1.1 and of each of the activity move rules (section 3.4.2). All tests are run on 20% of either the

dataset of Vanhoucke (2010) for the single–mode case or on 20% of both the MMLIB50 and MMLIB100 datasets for the multi–mode case.

3.6.2.1 Parameter testing

Table 3.5 shows the best found values for the parameters of our GA. As part of the finetuning of our GA we also tested different operators and found that the elite selection, one–point crossover and swap mutation for both the PL and ML performed best. It is important to stress that although both mutation rates and operators are the same, they are applied independently. This means that for a specific PL and ML, a swap is first applied to the PL and a different swap is subsequently applied to the ML, given both their mutation rates. Based on the tests for the proposed GA, it can be concluded that the results are in line with those of chapter 2 for the single–mode PAC model (Leyman and Vanhoucke, 2015) and that the GA used here is similar to the GA2 algorithm proposed by the authors, which displayed the best results to date for the RCPSPDC.

PAC/PP/PEO							**PAC**		**PP/PEO**	
$\|P_0\|$	$\|P\|$	$\|C\|$	$\|R\|$	M_1	M_2	MI	Y_1	Y_2	Y_1	Y_2
500	50	50	5	0.95	0.95	0.35	10,000	0.85	10,000	0.75

Table 3.5: Algorithm parameters.

Next, we consider the parameters of both penalty functions, namely Y_1, Y_2, Y_3 and Y_4. For Y_1 and Y_2 the best found values are displayed in table 3.5, whereas the values for Y_3 and Y_4 are shown in table 3.6. Based on the second table the following can be concluded:

- SM–PP/PEO: A higher profit margin leads to a higher value for the Y_4 parameter, with the exception of the case in which *%Prof* is 0%. This implies that for a higher (lower) average project NPV ($AvNPV$) a higher (lower) value for Y_4 is required.
- MM–PAC: Starting from a *%Neg* of 0% the values of Y_4 first decrease and subsequently increase as *%Neg* increases. The same trend can be observed in the absolute values of $AvNPV$, which means that a higher (lower) value for the Y_4 parameter is needed as the absolute value of the average project NPV increases (decreases).
- MM–PP/PEO: As the profit margin increases a higher Y_4 value is required until a profit margin of 40%, after which the Y_4 value decreases. No clear link can however be made between $AvNPV$ and the value of Y_4.

We test under what conditions the three mode improvement methods of section 3.4.1.1 are applied. The ERR reduction is always applied if the solution has a positive ERR,

	SM–PP/PEO			MM–PAC			MM–PP/PEO		
%Neg/%Prof	Y_3	Y_4	*AvNPV*	Y_3	Y_4	*AvNPV*	Y_3	Y_4	*AvNPV*
0%	20,000	0.75	-35.32	17,500	0.50	15,649.97	17,500	0.25	-238.68
20%	20,000	0.70	1,374.27	17,500	0.45	9,026.78	17,500	0.45	2,697.71
40%	20,000	0.70	2,899.61	17,500	0.35	2,471.52	17,500	0.50	5,678.73
60%	20,000	0.80	4,433.25	17,500	0.35	-855.54	17,500	0.50	8,690.59
80%	20,000	0.85	5,959.59	17,500	0.40	-5,623.10	17,500	0.40	11,654.86
100%	20,000	0.995	7,494.97	17,500	0.50	-14,304.44	17,500	0.20	14,665.94

Table 3.6: Parameters penalty function (3.13).

independent of the resource characteristics, whereas the CP and WC improvement methods are used if the ERR is equal or has been reduced to zero. Specific conditions apply for the CP and WC methods depending on the renewable and non–renewable resource strength, as can be seen in table 3.7.

		RS^{ρ}		
		0.25	0.50	0.75
RS^{ν}	0.25		CP	CP
	0.50	WC	CP+WC	CP
	0.75	WC	CP+WC	CP

Table 3.7: Applicability improvement methods.

The conditions under which the minimal sum of durations or minimal total work content is chosen to evaluate the initial $|P_0|$ random ML also depend on the resource strength. The sum of durations is selected if the problem instance under consideration's RS^{ρ} is larger than 0.25, whereas the total work content is employed otherwise.

3.6.2.2 Mode improvement

Table 3.8 shows the added value of penalty function (3.12) for the ERR violations, of the three mode improvement steps of section 3.4.1.1 (ERR reduction, CP improvement and WC improvement) and of the schedule improvement applied after the SSGS. Each of the three parts of our algorithm are omitted (*NoPenFunc*, *NoModeChange* and *NoSchedImpr*) and compared with the results of the method without any omissions (*Full*). The results are evaluated based on the percentage of *D–Feas* solutions found (*%D*) and based on the percentage of *ERR–Feas* solutions found (*%E*). For *%D* any solutions which are *D–Feas* but *ERR–Infeas* have been counted as *D–Infeas* to ensure only feasible solutions with respect to both the RR and NRR are taken into account for *%D*, since deadline infeasibility can be caused by the PL, ML or both. *%E* on the contrary takes *ERR–Feas* solutions into account regardless of deadline feasibility since the ERR value is only determined by the

ML. The following can be concluded from table 3.8:

- For all three payment models, penalty function (3.12) has a clear added value, since not applying it (*NoPenFunc*) reduces the *%E* and the *%D*. Whereas the decrease in *%E* was expected in advance the decrease in *%D* shows that due to the interaction with penalty function (3.13), function (3.12) also helps to improve the deadline feasibility.
- Without the mode improvement steps (*NoModeChange*) the performance of our algorithm is considerably worse both in terms of *%D* and *%E*.
- Omitting the schedule improvement (*NoSchedImpr*) leads to a decrease in *%D*, whereas *%E* remains largely unchanged. The decrease in *%D* is furthermore larger than if the mode improvement steps are not included (*NoModeChange*).

		Full		**NoPenFunc**		**NoModeChange**		**NoSchedImpr**	
		%D	*%E*	*%D*	*%E*	*%D*	*%E*	*%D*	*%E*
PAC	50	91.32	100.00	83.64	97.18	78.55	91.13	83.45	99.92
	100	86.11	100.00	78.82	99.85	58.10	85.34	76.62	100.00
PP	50	93.56	100.00	83.02	96.80	86.54	90.01	77.62	99.85
	100	86.88	100.00	75.42	98.61	71.99	83.22	62.08	99.92
PEO	50	91.74	100.00	82.18	97.18	82.95	87.89	74.50	99.85
	100	84.91	100.00	73.77	98.80	67.75	83.10	57.91	100.00

Table 3.8: Added value of mode improvement.

3.6.2.3 Activity move rules

Table 3.9 displays the added value of both the network– and schedule–based activity move rules. In the table we compare the average percentage improvement of several cases with the results if no activity move rules at all are applied. We compare the proposed scheduler (*S1–2*) with the results if only the network–based moves are used (*S1*), if only the schedule–based moves are included (*S2*) and if both steps are swapped (*S2–1*). The latter case implies that instead of first applying the network–based moves and then the schedule–based delays, we now first employ the schedule–based moves and only use the network–based moves afterwards. For the PP and PEO models we test the additional case if the potential NPV calculations and moves are omitted ($NoNPV_{pot}$). Finally, only problem instances for which all methods could find a *D–Feas* solution are taken into account in the table.

First, we discuss the results for the single–mode (SM) PP and PEO models. Based on table 3.9 it is clear that applying any form of activity moves is guaranteed to improve the project NPV, but that some options have a higher added value than others. The proposed methodology seen under *S1–2* clearly performs best for both the PP and PEO model. The schedule–based moves have a strong added value whereas if only the network–based moves would be applied, this would lead to the worst results of the five options in table 3.9. The added value of the potential NPV calculations can be seen by comparing the last column with the previous two columns. Without the potential NPV the added value of the activity move rules would be lower.

Second, we focus on the results for the multi–mode (MM) cases of the PAC, PP and PEO models. For the PAC model the results of *S1–2* are best, but *S1* is a close second. The schedule–based moves have a smaller added value, which is in line with the results for the single–mode PAC as reported in chapter 2 (Leyman and Vanhoucke, 2015). For the multi–mode PP and PEO models the largest added value is again created by the network–based moves (*S1*), which is in stark contrast with the single–mode cases where the schedule–based rules have the largest added value. The contribution of the potential NPV can be seen by comparing the final three columns, which show that the results would not be as good without the potential NPV. Finally, the best results are achieved by *S1–2*.

All the average percentage improvements in the table reported a p–value smaller than 0.01, when tested for statistical significance, with both a one–way ANOVA and paired samples t–tests (*).

		S1	*S2*	*S1–2*	*S2–1*	$NoNPV_{pot}$
PAC	MM50	3.89*	3.65*	3.90*	3.87*	/
	MM100	6.22*	6.14*	6.23*	6.19*	/
PP	SM	5.56*	6.53*	6.70*	6.65*	6.14*
	MM50	9.17*	8.48*	9.26*	9.23*	9.08*
	MM100	8.86*	7.84*	8.92*	8.85*	8.57*
PEO	SM	9.77*	10.99*	12.38*	12.14*	10.37*
	MM50	7.19*	6.28*	7.22*	7.13*	7.03*
	MM100	6.75*	5.52*	6.82*	6.79*	6.61*

Table 3.9: Added value of activity move rules (average % improvement).

3.6.3 Best known results

In this section, we show the final results of our algorithm for the single– and multi–mode cases based on tests for the entire datasets. Table 3.10 displays the results for the single–mode PP and PEO models, and also includes the results of chapter 2 for the PAC

model (Leyman and Vanhoucke, 2015) for the sake of completeness. Table 3.11 shows the results for the multi–mode PAC, PP and PEO models for MMLIB50, MMLIB100 and MMLIB+ respectively. Since the MMLIB data however is relatively new, we also employ the PSPLIB data of Kolisch and Sprecher (1996) (table 3.12). The results are evaluated based on two criteria, namely the average project NPV ($AvNPV$) of all feasible solutions, and the percentage *D–Feas* solutions ($\%D$). All reported results are *ERR–Feas* as a result of the ERR improvement method and of penalty function (3.12).

Based on the results in table 3.10 for the single–mode cases the following can be concluded:

- Increases in *Act* increase $AvNPV$ and decrease the deadline feasibility for all three models, although the decrease in $\%D$ is limited which reflects favorably on the methods used.

- An increase in *D–Incr* improves $AvNPV$ but only for PAC, whereas the deadline feasibility increases for all three payment models.

		PAC		**PP**		**PEO**	
		$AvNPV$	$\%D$	$AvNPV$	$\%D$	$AvNPV$	$\%D$
Act	25	290.13	99.51	2,067.45	99.77	2,036.09	99.75
	50	978.02	99.03	3,514.95	99.28	3,467.58	99.49
	75	1,648.44	98.06	4,376.94	98.31	4,322.15	98.08
	100	2,014.01	97.52	4,980.82	97.25	4,915.11	97.06
D–Incr	5%	986.13	94.21	3,610.71	94.65	3,569.77	94.40
	10%	1,179.56	99.93	3,761.39	99.95	3,703.56	99.98
	15%	1,290.03	100.00	3,762.73	100.00	3,709.23	100.00
	20%	1,440.25	100.00	3,758.98	100.00	3,708.81	100.00

Table 3.10: Best known results single–mode.

Based on the results in tables 3.11 and 3.12 for the multi–mode cases the following can be concluded:

- A higher number of activities decreases deadline feasibility, similar to the single–mode results. The decrease in $\%D$–*Feas* is lowest for the PP and PEO models on the PSPLIB and MMLIB50 data, but largest on the MMLIB100 and MMLIB+ data. A larger decrease between the MMLIB50 and MMLIB100 results on the one hand and the MMLIB+ results on the other hand can be observed. This implies that the proposed algorithm scales well as *Act* increases, but that it has more difficulty finding *D–Feas* solutions with a greater number of resources and activity modes.

- Increases in *D–Incr* considerably increase *%D* especially for the MMLIB data. An increase in *%D* has a positive effect on *AvNPV* for PSPLIB. The effect of *AvNPV* is not clear for MMLIB, though this may be due to more instances becoming *D–Feas* with an increase in project deadline. This would imply that these, in terms of *D–Feas* more difficult, instances in general have a lower NPV.

		PAC		**PP**		**PEO**	
	D–Incr	*AvNPV*	*%D*	*AvNPV*	*%D*	*AvNPV*	*%D*
MMLIB50	5%	839.21	76.14	5,259.72	79.10	5,167.08	78.83
	10%	655.95	93.64	5,166.80	97.93	5,034.61	97.38
	15%	545.35	98.58	5,154.78	99.78	5,059.46	99.69
	20%	568.48	99.72	5,159.60	99.91	5,047.04	99.94
MMLIB100	5%	2,696.03	62.65	9,943.92	58.95	9,634.86	59.07
	10%	2,285.26	86.98	9,637.29	90.40	9,450.83	90.00
	15%	2,078.43	95.93	9,576.37	98.49	9,381.59	98.18
	20%	1,905.87	99.07	9,529.43	99.69	9,317.80	99.81
MMLIB+	5%	2,002.44	35.46	6,472.98	20.75	6,323.49	20.80
	10%	2,016.36	68.55	6,177.09	55.27	6,066.63	55.35
	15%	1,669.42	87.14	6,177.92	79.53	6,043.29	79.64
	20%	1,470.68	95.49	6,239.32	90.88	6,109.93	91.11

Table 3.11: Best known results MMLIB.

In table 3.11 we see that the deadline feasibility is relatively low for MMLIB compared to the single–mode results of table 3.10, especially when *D–Incr* is 5%. As such, we have decided to test our methodology for the maximization of project NPV on the makespan minimization variant of the MRCPSPDC, namely the MRCPSP, by omitting all activity move rules and by setting the project deadline to the best known solution to date (i.e. *D–Incr* is set to 0%). The penalty functions' parameters are those used for MM–PAC with *%Neg* equal to 0%.

In table 3.13 we compare our results with the best known results from literature based on the average percentage deviation from the critical path based lower bound for the MMLIB50, MMLIB100 and MMLIB+ datasets. The percentage between brackets for the paper of Van Peteghem and Vanhoucke (2010) means that the algorithm proposed in the paper was only able to find *ERR–Feas* solutions in 97.59% of the cases, whereas all other algorithms shown always found a feasible solution. Based on the results from the table it should be clear that whereas Van Peteghem and Vanhoucke (2011) still have the best results to date, our algorithm consistently performs second best. This implies that our makespan minimization methods and penalty functions still perform well, but shows that it is considerably more difficult to reach strict deadlines, as *Act* increases and as a higher number of activity modes and resources is used. The results in table 3.13 also display

		PAC		**PP**		**PEO**	
	D–Incr	*AvNPV*	*%D*	*AvNPV*	*%D*	*AvNPV*	*%D*
J10	5%	399.13	98.41	1,126.29	99.60	1,096.03	99.63
	10%	415.02	99.81	1,135.44	100.00	1,108.88	100.00
	15%	423.39	100.00	1,136.28	100.00	1,113.56	100.00
	20%	433.14	100.00	1,137.94	100.00	1,113.59	100.00
J12	5%	245.32	97.78	1,362.20	99.36	1,331.69	99.18
	10%	257.01	99.70	1,367.21	100.00	1,334.19	100.00
	15%	268.91	100.00	1,369.01	100.00	1,341.45	100.00
	20%	282.19	100.00	1,370.27	100.00	1,341.35	100.00
J14	5%	315.27	96.55	1,431.11	98.67	1,390.00	98.55
	10%	316.70	99.82	1,436.33	100.00	1,399.34	99.94
	15%	328.90	99.97	1,439.65	100.00	1,411.30	99.94
	20%	343.23	100.00	1,441.60	100.00	1,411.44	100.00
J16	5%	271.83	94.18	1,519.76	98.88	1,484.50	98.36
	10%	265.97	98.67	1,530.88	100.00	1,492.82	100.00
	15%	278.21	99.70	1,534.98	100.00	1,500.50	100.00
	20%	294.11	99.91	1,537.51	100.00	1,502.23	100.00
J18	5%	159.62	94.20	1,654.53	98.07	1,616.84	97.74
	10%	146.66	98.82	1,658.87	99.91	1,619.76	99.88
	15%	164.12	99.46	1,663.67	100.00	1,631.59	100.00
	20%	183.14	99.76	1,665.74	100.00	1,629.60	100.00
J20	5%	309.22	93.02	1,876.57	97.32	1,834.20	97.26
	10%	273.81	98.83	1,876.76	99.76	1,836.98	99.85
	15%	286.32	99.88	1,880.75	100.00	1,842.36	100.00
	20%	308.06	99.82	1,882.68	100.00	1,842.76	100.00
J30	5%	-272.39	85.60	2,832.14	91.55	2,772.72	91.82
	10%	-344.44	97.89	2,834.11	99.70	2,765.70	99.76
	15%	-341.33	99.58	2,837.21	99.97	2,775.68	100.00
	20%	-305.19	100.00	2,841.30	100.00	2,784.58	100.00

Table 3.12: Best known results multi–mode PSPLIB.

the percentage of instances for which the best known solution to date is found by our algorithm (*%BestFound*) and the percentage of instances for which our results improved upon the best known solutions (*%NewBest*).

We use a multivariate linear regression analysis to clearly and statistically show the effect of the project parameters on the objective function. The results are reported in table 3.14 which provides the R^2–value, the constant (*Const*) and the coefficient for each parameter. An asterisk (*) is used to indicate that a coefficient is significant at the 1% confidence level.

With respect to the data parameters, the following conclusions can be drawn:

- **Act:** An increase in the number of activities reflects favorably on *AvNPV*. Although the coefficients are rather small for all six models, they are all significant at the 1%

	MMLIB50	MMLIB100	MMLIB+
Van Peteghem and Vanhoucke (2011)	**25.45**	**26.51**	**101.45**
This chapter	26.36	27.88	106.37
Van Peteghem and Vanhoucke (2010)	27.12	29.55	(*97.59*)
Lova et al. (2009)	28.59	31.01	114.07
Damak et al. (2009)	32.46	36.87	126.69
Józefowska et al. (2001)	33.81	39.05	121.09
%BestFound This chapter	50.93	40.19	11.11
%NewBest This chapter	0.37	0.37	0.68

Table 3.13: Solutions makespan minimization multi–mode.

level of confidence.

- **OS:** An increase in *OS* greatly and significantly decreases project NPV for all six models. In more parallel networks *AvNPV* is larger because more activities can be scheduled in parallel. Specifically, for the PAC model more cash inflows (outflows) can be scheduled earlier (later). In the PP and PEO models the activities can be scheduled closer to payment times, which reduces the NPV of cash outflows, and more activities can be completed at earlier payment times, which increases the NPV of cash inflows.

- **RU/RF:** The RU (data of Vanhoucke (2010)) and RF (MMLIB) have a strong negative effect on project NPV, both for the single– and multi–mode parameters (RF^{ρ} and RF^{ν}). The larger the average number of project resources used the lower *AvNPV*, except for the SM–PAC case.

- **RC/RS:** In terms of the average amount of resource usage it can be observed that a higher value for *RC*, which implies a higher average resource usage, has a negative impact on the project NPV for the SM–PP and SM–PEO cases, whereas the impact is positive for the SM–PAC model. The coefficient of the latter is however not significant at the 1% confidence interval. In terms of the *RS* parameter values (RS^{ρ} and RS^{ν}) a similar trend can be observed: a higher resource usage leads to a lower average NPV. Take into account however that, unlike for *RC*, increases in the value of the *RS* parameters imply a lower resource usage. As such the coefficients for the multi–mode cases which use *RS* instead of *RC* have a positive sign.

- **D–Incr:** For the PAC model an increase in the project deadline leads to a significantly higher *AvNPV*, although the effect is limited in size. For the PP and PEO models on the contrary no significant effect exists for the single–mode cases. In the multi–mode cases an increase in *D–Incr* slightly decreases the project NPV.

- **%Neg/%Prof:** As could be expected, increases in %Neg (%Prof) lead to a lower (higher) $AvNPV$. All six coefficients are significant at the 1% confidence interval.

	SM–PAC	**SM–PP**	**SM–PEO**	**MM–PAC**	**MM–PP**	**MM–PEO**
R^2	0.771	0.768	0.768	0.853	0.883	0.883
Const	5,966.77*	-132.74	-155.62	10,531.86*	-3,186.76*	-3,263.21*
Act	23.29*	38.51*	37.99*	28.72*	79.09*	77.32*
OS	-1,540.60*	-1,191.89*	-1,182.57*	-929.89*	-3,371.26*	-3,266.13*
RU/RF^{ρ}	132.13*	-541.47*	-535.79*	-207.79*	-1,617.20*	-1,522.37*
RC/RS^{ρ}	121.59	-371.59*	-369.97*	614.78*	1,859.05*	1,799.46*
RF^{ν}	/	/	/	-220.52*	-919.12*	-919.45*
RS^{ν}	/	/	/	479.03*	1,191.05*	1,160.87*
D–Incr	30.95*	4.93	5.50	27.97*	-9.91*	-8.12*
%Neg/%Prof	-125.58*	76.21*	75.63*	-234.47*	130.16*	129.16*

Table 3.14: Results multivariate regression.

Since the reported R^2 values lie between 0.771 and 0.883 the proposed models can predict the project NPV well based on the problem characteristics. Given that most coefficients are furthermore significant at the 1% confidence interval, the models can be used as predictors for our algorithm's performance in terms of $AvNPV$, and clearly show the effect of the data parameters on the objective function.

3.7 Conclusions

In this chapter, we discussed the resource–constrained project scheduling problem with discounted cash flows (RCPSPDC) and its multi–mode variant the MRCPSPDC for multiple payment models. First, we have extended an existing scheduling technique to make it applicable in all of the six cases used in the manuscript. To do so, we have discussed the importance of peaks in the activity profit curves for optimizing the activity schedule with different payment models. The activity move rules have been applied in two ways, namely based on the network predecessors and successors, and based on the neighboring activities in the project schedule. These rules move a set of activities together in order to improve project net present value (NPV). We have introduced the notion of potential NPV to delay certain activities in the set beyond the delay for the entire set. Two penalty functions have been applied to improve project feasibility with respect to both the project deadline and the mode selection feasibility. Furthermore, several mode improvements from literature have been used as part of a proposed genetic algorithm metaheuristic. Second, the proposed solution methodology has been extensively tested on several existing datasets and conclusions have been drawn with respect to the influence of the data's parameters

on the project NPV. Finally, the results in this chapter can be used as future benchmarks for each of the six models discussed.

In the future, it may be worthwhile to propose local searches, which simultaneously delay activities and make mode changes, in order to improve NPV in two ways. In doing so, explicit rules to improve the project NPV by applying mode changes, could be incorporated in the overall framework discussed in this chapter. Another future research avenue concerns the proposed penalty functions. Different types of functions could be investigated in detail, whereas the integration and generalization of the two functions discussed in this chapter, may increase performance.

4

Metaheuristics for the discrete time/cost trade–off problem with net present value optimization and different payment models

In this chapter, we focus on the deadline version of the discrete time/cost trade–off problem with net present value optimization (DTCTP–NPV). We apply three payment models from literature, which all assume that cash outflows occur at activity completion and that cash inflows depend on the actual project schedule. Our contribution to existing literature is threefold. First, we introduce a full activity–on–the–node mathematical model for the DTCTP–NPV with the three payment models. Care has been taken to model the problem in line with the time indexed binary decision variables commonly used in literature. Second, we propose two solution representations as part of a metaheuristic to solve the DTCTP–NPV with the three payment models. Third, we compare the proposed methodologies and show that our results significantly outperform existing work, based on an extensive computational experiment.

4.1 Introduction

Time/cost trade–off problems have been extensively discussed in literature and focus on the trade–off between an activity's duration and its associated cost. In the discrete variant of the problem it is assumed that the trade–off follows a discrete non–increasing pattern, which implies that an activity's duration can be shortened by incurring a higher activity cost. Three different problem formulations can be distinguished in literature. The first is called the deadline problem and focusses on minimizing the total project cost given a fixed deadline. The second variant is the budget problem which aims to minimize the project duration without exceeding a cost threshold. A third problem constructs the complete and efficient frontier in terms of time/cost combinations for all feasible project durations. For overviews of the discrete time/cost trade–off problem, we refer to Vanhoucke and Debels (2007) and Hazir et al. (2010). In the remainder of this chapter, we go into detail about an extension of the deadline variant of the DTCTP, namely the DTCTP with net present value (NPV) optimization (DTCTP–NPV), for three payment models. From a methodology perspective, we focus on **solution representations** and **schedulers** for these representations as part of a metaheuristic.

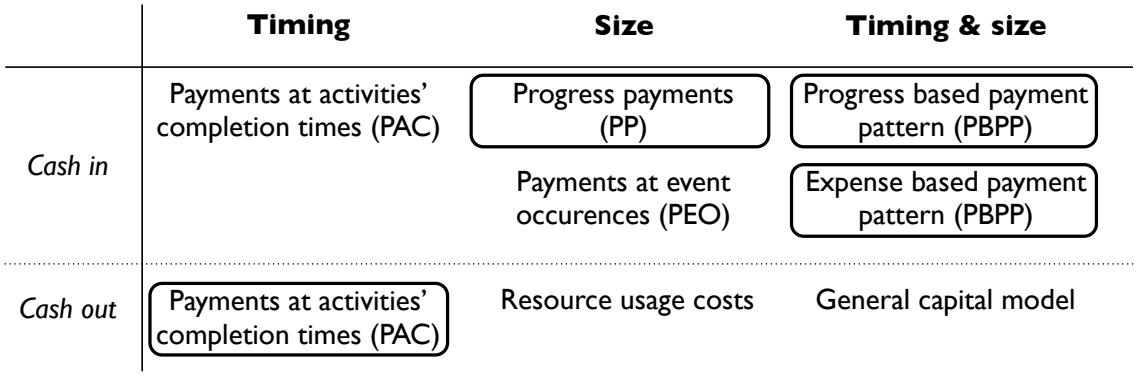


Figure 4.1: Overview of the research on project scheduling with NPV optimization in chapter 4.

The DTCTP–NPV has first been discussed by Erengüç et al. (1993). Cash flows are associated with events and the authors illustrate that the DTCTP–NPV is a combination of both the DTCTP and the payment scheduling problem. Vanhoucke and Debels (2007) extend upon the work of Erengüç et al. (1993) and assume cash outflows are associated with arcs and cash inflows with events as part of an activity–on–the–arc (AoA) representation. The authors propose a metaheuristic algorithm which succeeds in finding near–optimal solutions for the DTCTP–NPV. He and Xu (2008) discuss the DTCTP–NPV with a bonus–penalty structure and include a trade–off mechanism between the contractor and the client. Their model assigns cash in– and outflows to event occurrence times. The objective is to

determine both activity completion times and payment times to balance the NPV of both parties. A simulated annealing metaheuristic is proposed, which is tested on an example from literature. The client's perspective is analyzed by He et al. (2009a) by making use of a simulated annealing algorithm similar to the authors' previous approach. He et al. (2009b) extend their earlier work by introducing three types of payment models on top of the previously used general model, and study the problem from the contractor's point of view. It is concluded that the simulated annealing algorithm performs best out of several alternatives. He et al. (2012) further extend the problem formulation by including a capital constraint for the models discussed, which ensures that the cash position of the contractor can at no point in the project schedule become negative. The results show that the loop nested tabu search algorithm leads to the best results. Last, He et al. (2014) apply their simulated annealing algorithm to the DTCTP–NPV with the inclusion of financing costs. The goal is to distribute these financing costs among the contractor and client in a manner acceptable for both parties.

In this chapter, we focus on the DTCTP–NPV from the contractor's perspective with the three different payment models of He et al. (2009b). We propose two metaheuristics to solve the problem. The remainder of this chapter is organized as follows. In section 4.2, the problem definition of the DTCTP–NPV is discussed, whereas section 4.3 focuses on two solution representations. Section 4.4 provides details of our proposed metaheuristic. In section 4.5, the results of our algorithm are analyzed and compared with the method of He et al. (2009b). We finish with a conclusion and recommendations for future research in section 4.6.

4.2 Problem definition

In this section, the problem definition of the DTCTP–NPV is highlighted for each of the three employed payment models of He et al. (2009b). We furthermore use an example to illustrate the three payment models.

In general, a project can be represented by a network or directed graph $G(N, A)$ with N the nodes or project activities and A the arcs of precedence relations between individual activities. In this chapter, we use the activity–on–the–node (AoN) representation and assume a finish–start time–lag of zero for the precedence relations, which implies an activity can be started once all of its predecessors have been completed. The activities i ($i \in N = \{1, \ldots, n\}$) each have a mode m with $m \in M_i = \{1, \ldots, |M_i|\}$. Each of these modes m of an activity i has a duration d_{im} and a cost c_{im}. These activity durations are discrete non–increasing functions of the costs associated with them, i.e. $d_{i1} > d_{i2} > \ldots > d_{i|M_i|}$ and $c_{i1} < c_{i2} < \ldots < c_{i|M_i|}$. Additionally, a start dummy activity 0 and an end dummy

activity $n+1$ are included, but both activities only have one mode with a duration and cost of zero. The project has a deadline equal to δ_{n+1}.

The deadline variant of the discrete time/cost trade–off problem with NPV maximization aims to optimize the project NPV subject to precedence constraints. The project NPV consists of two parts, namely the earlier defined cash outflows or costs c_{im} of an activity i executed in mode m, and the additionally introduced cash inflows c_i^+ for each activity i, which are independent of the selected modes. The cash outflows are discounted to activity completion and are accrued in a linear manner starting from the activity start time, whereas the receipt of cash inflows depends on the payment model used.

Table 4.1 provides an overview of the notations used and explained in the following sections.

Notation	Definition
A	Set of arcs
N	Set of activities
i	Activity index
M_i	Set of modes of activity i
m	Mode index
P_i	Set of immediate predecessors of activity i
S_i	Set of immediate successors of activity i
ef_i	Earliest finish time of activity i
lf_i	Latest finish time of activity i
s_i	Available slack of activity i, i.e. $s_i = lf_i - ef_i$
d_{im}	Duration of activity i with mode m
c_{im}	Cost/cash outflow of activity i with mode m (contractor)
c_i^+	Cash inflow of activity i (contractor)
p_t	Payment amount received at time t (contractor)
v_i	Total created value of activity i (client)
cv_{imtw}	Created value of activity i with mode m and finish time t, at time w (client)
θ	Compensation proportion
K	Number of payments
B	Benchmark for project costs
δ_{n+1}	Project deadline
α	Discount rate

Table 4.1: Overview of notations.

4.2.1 Progress–based payment pattern

The progress–based payment pattern (PBPP) assumes payments are arranged according to the progress in the project schedule. In this payment model the project client, or party receiving the benefits of the project, pays the contractor, or party responsible for executing the project, according to the work done in terms of created value for the

client. The total created value for the client and the total reimbursement or cash inflow for the contractor, are furthermore linked by a compensation proportion θ ($0 < \theta < 1$): $\sum_{i=1}^{n} c_i^+ = \theta \cdot \sum_{i=1}^{n} v_i$, with v_i the total created value by activity i. A threshold value, which determines when payments are received by the contractor, is introduced and equals $\sum_{i=1}^{n} v_i/K$, with K the number of payments. The number of payments is determined in advance. This reasoning ensures that once the accumulated created value since the previous payment exceeds this threshold value, a payment p_t is received. This payment equals the total created value until the payment time multiplied with θ and minus any previous payments. The final payment is received at the project deadline. The following can be concluded with respect to the cash flows of the contractor and the client:

- Client: a created value $\sum_{i=1}^{|N|} v_i$ is generated by the project, independent of the contractor's actions. The NPV of this value, on the contrary, depends on the finish times of the activities, which are based on the contractor's schedule.

- Contractor: a cost is incurred for each activity, which depends on the selected activity modes by the contractor. The activity costs are discounted to the activity finish times. A portion θ of the total created value is received from the client as compensation for the work done. The size and timing of payments depend on the work completed. This way, the value of θ determines the profitability of the project for the contractor. In this chapter, we focus on the contractor's perspective and optimize this party's NPV.

As stated, the progress of the project is measured at any time unit w during the project duration based on the created value until that point in the schedule. This translates into a created value of $\sum_{m=1}^{|M_i|} \sum_{t=ef_i}^{lf_i} cv_{imtw} \cdot x_{imt}$ for each activity i, with the parameter cv_{imtw} the created value of activity i executed in mode m with a finish time of t, at time w. The value of this parameter cv_{imtw} can be calculated in advance for all activity, mode and finish time combinations for each time unit w until the project deadline:

1. Activity i has not yet been started by time w: $\sum_{m=1}^{|M_i|} \sum_{t=ef_i}^{lf_i} cv_{imtw} \cdot x_{imt} = 0$.
2. Activity i is in progress at time w: $0 < \sum_{m=1}^{|M_i|} \sum_{t=ef_i}^{lf_i} cv_{imtw} \cdot x_{imt} < v_i$.
3. Activity i has been completed by time w: $\sum_{m=1}^{|M_i|} \sum_{t=ef_i}^{lf_i} cv_{imtw} \cdot x_{imt} = v_i$.

In our model we consider the traditional time indexed binary decision variables x_{imt} (Kolisch and Drexl, 1997; Pritsker et al., 1969), and include additional variables y_t to model the occurrence time of payments.

$$x_{imt} = \begin{cases} 1 & \text{if activity } i \text{ is executed in mode } m \text{ and finishes at time } t \\ 0 & \text{otherwise} \end{cases}$$

$$y_t = \begin{cases} 1 & \text{if payment is received at time } t \\ 0 & \text{otherwise} \end{cases}$$

A mathematical model for the DTCTP–NPV with the PBPP is constructed as follows:

$$\text{Maximize} \sum_{t=1}^{\delta_{n+1}} p_t \cdot e^{-\alpha t} - \sum_{i=1}^{|N|} \sum_{m=1}^{|M_i|} \sum_{t=ef_i}^{lf_i} c_{im} \cdot e^{-\alpha t} \cdot x_{imt} \tag{4.1}$$

Subject to:

$$\sum_{m=1}^{|M_i|} \sum_{t=ef_i}^{lf_i} x_{imt} = 1, \quad \forall i \in N, \tag{4.2}$$

$$\sum_{m=1}^{|M_j|} \sum_{t=ef_j}^{lf_j} t \cdot x_{jmt} \leq \sum_{m=1}^{|M_i|} \sum_{t=ef_i}^{lf_i} (t - d_{im}) \cdot x_{imt}, \quad \forall i \in N;\ \forall j \in P_i, \tag{4.3}$$

$$\sum_{t=ef_{n+1}}^{lf_{n+1}} t \cdot x_{(n+1)1t} \leq \delta_{n+1}, \tag{4.4}$$

$$p_w = \left(\theta \cdot \sum_{i=1}^{|N|} \sum_{m=1}^{|M_i|} \sum_{t=ef_i}^{lf_i} cv_{imtw} \cdot x_{imt} - \sum_{t=0}^{w-1} p_t \right) \cdot y_w, \quad w = 1, \ldots, \delta_{n+1}, \tag{4.5}$$

$$\sum_{i=1}^{|N|} \sum_{m=1}^{|M_i|} \sum_{t=ef_i}^{lf_i} cv_{imtw} \cdot x_{imt} - \sum_{i=1}^{|N|} v_i/K \cdot \sum_{t=0}^{w-1} y_t \geq \sum_{i=1}^{|N|} v_i/K \cdot y_w - M \cdot (1 - y_w), \quad w = 1, \ldots, \delta_{n+1}, \tag{4.6}$$

$$\sum_{i=1}^{|N|} \sum_{m=1}^{|M_i|} \sum_{t=ef_i}^{lf_i} cv_{imtw} \cdot x_{imt} - \sum_{i=1}^{|N|} v_i/K \cdot \sum_{t=0}^{w-1} y_t < \sum_{i=1}^{|N|} v_i/K \cdot (1 - y_w) + M \cdot y_w, \quad w = 1, \ldots, \delta_{n+1}, \tag{4.7}$$

$$\sum_{t=0}^{\delta_{n+1}} p_t = \theta \cdot \sum_{i=1}^{|N|} v_i, \tag{4.8}$$

$$\sum_{t=0}^{\delta_{n+1}} y_t = K, \tag{4.9}$$

$$y_0 = 0, \quad p_0 = 0, \tag{4.10}$$

$$y_{\delta_{n+1}} = 1, \tag{4.11}$$

$$x_{imt} \in \{0, 1\}, \quad \forall i \in N;\ m \in M_i;\ t = ef_i, \ldots, lf_i, \tag{4.12}$$

$$y_t \in \{0, 1\}, \quad t = 0, \ldots, \delta_{n+1} \tag{4.13}$$

The objective function (4.1) maximizes the project NPV based on a discount rate α. The payments p_t constitute the cash inflows for the model, whereas the cash outflows

c_{im} depend on the selected activity modes and are discounted to the activity completion time. Constraints (4.2) ensure that every activity is executed in exactly one mode and has exactly one finish time, whereas constraints (4.3) enforce the precedence relations between activities. The project deadline is included in constraint (4.4). Constraints (4.5) determine the size of a payment based on the compensation proportion θ and on the total created value at time w. Constraints (4.6)–(4.7) are big M constraints (with M a very large positive integer), which model the occurrence time of payments included in the variables y_w and which are based on the threshold value of $\sum_{i=1}^{n} v_i/K$. The total monetary value of the payments received has to be in line with θ and the created value for the client (constraint (4.8)), and the number of payments has to equal K (constraint (4.9)). Constraints (4.10) model that no payment can occur at time 0, and constraint (4.11) ensures that the final payment is received at the project deadline. The binary variable constraints are included in (4.12)–(4.13).

The PBPP can be represented as $1, T|cpm, \delta_n, disc, mu, sched|npv$ according to the classification scheme of Herroelen et al. (1999), and as $MPS1|prec|\sum C_i^F \beta^{C_i}$ following Brucker et al. (1999).

4.2.2 Expense–based payment pattern

The expense–based payment pattern (EBPP) focusses on refunding the expenses of the contractor. Payments are received every time the total expenses of the contractor exceed a threshold. This threshold is calculated as B/K with B the agreed upon benchmark for the project costs and K the number of payments. B is calculated as the sum of the average mode cost for all activities: $B = \sum_{i=1}^{|N|}(\sum_{m=1}^{|M_i|} c_{im}/|M_i|)$. The model of the PBPP in section 4.2.1 can be used for the EBPP as well, with the only difference that constraints (4.6) and (4.7) should be replaced with constraints (4.14) and (4.15), to account for the difference in payment threshold between both payment models.

$$\sum_{i=1}^{|N|}\sum_{m=1}^{|M_i|}\sum_{t=ef_i}^{w} c_{im}\cdot x_{imt} - B/K\cdot\sum_{t=1}^{w-1} y_t \geq B/K\cdot y_w - M\cdot(1-y_w), \quad w=1,\ldots,\delta_{n+1} \tag{4.14}$$

$$\sum_{i=1}^{|N|}\sum_{m=1}^{|M_i|}\sum_{t=ef_i}^{w} c_{im}\cdot x_{imt} - B/K\cdot\sum_{t=1}^{w-1} y_t < B/K\cdot(1-y_w) + M\cdot y_w, \quad w=1,\ldots,\delta_{n+1} \tag{4.15}$$

The EBPP can be represented as $1, T|cpm, \delta_n, disc, mu, sched|npv$ according to the classification scheme of Herroelen et al. (1999), and as $MPS1|prec|\sum C_i^F \beta^{C_i}$ following Brucker et al. (1999).

4.2.3 Time–based payment pattern (progress payments)

The third model is the time–based payment pattern (TBPP), in which payments occur once a specified amount of time has passed in the project. The threshold is set to $\lceil \delta_{n+1}/K \rceil$, with K the number of payments. A simplified mathematical model can be applied since the payment times are known in advance and independent of the project schedule, which is not the case for the PBPP and EBPP. Specifically, constraints (4.5)–(4.7) can be omitted as well as the y_w decision variables. p_w equals $\sum_{i=1}^{|N|} \sum_{m=1}^{|M_i|} \sum_{t=ef_i}^{lf_i} cv_{imtw}$ if time w is a payment time and is set to zero otherwise. Just like for the PBPP and EBPP the size of the payments p_w still depends on the schedule under consideration and on previous payments made, but the binary decision whether time w is a payment time no longer depends on the project schedule. As a result, the TBPP corresponds with the progress payments (PP) model discussed in chapter 3.

The TBPP can be represented as $1, T|cpm, \delta_n, disc, mu, per|npv$ according to the classification scheme of Herroelen et al. (1999), and as $MPS1|prec|\sum C_i^F \beta^{C_i}$ following Brucker et al. (1999).

4.2.4 Example

To illustrate the three payment models, we use the example project shown at the top of figure 4.2. We assume a discount rate of 1%, a project deadline of 14, a compensation proportion of 0.9 and three payments. Based on the data in the figure it can be concluded that the total created value is 3,360 (= 300 + 360 + 660 + 900 + 540 + 600) and that the total cash inflow for the contractor equals 3,024 (= 0.9 x 3,360).

The project schedule used for the three models is shown at the bottom of figure 4.2 and uses the following mode list: (1, 1, 1, 2, 2, 1). Since the costs incurred by the contractor only depend on the modes selected for the activities, these costs are independent of the payment model used. The total costs of the schedule in figure 4.2 equal 1,370 (= 140 + 100 + 280 + 280 + 240 + 330) and the NPV of the costs is 1,247.22 (= 135.86 + 94.18 + 261.07 + 258.47 + 210.74 + 286.89). The total created value ($\sum v_i$) and total costs ($\sum c_{im}$) incurred at every time unit during the project duration are shown beneath the schedule in the figure.

- PBPP: Based on the total created value and the number of payments K, the payment threshold is set to 1,120 (= 3,360/3). As can be seen from figure 4.2 this implies that the first payment is received at time 6 and equals 1,039.50 or 0.9 x 1,155 (1,155 = 300 + 360 + 660 x 0.75). The second and third payments are received at times 8 and 14, and equal 1,055.70 (= 0.9 x (300 + 360 + 660 + 900 + 540 x 0.2) - 1,039.50) and

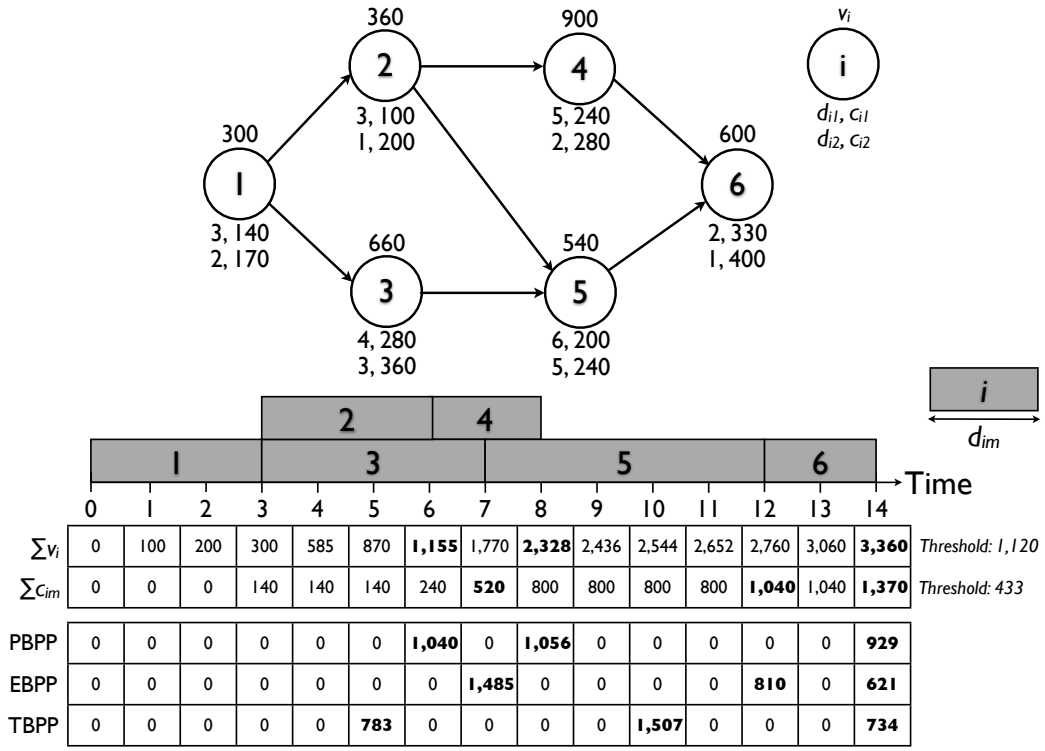
928.80 (= 0.9 x 3,360 - 1,039.50 - 1,055.70) respectively. These three time instances are selected as payment times since the total created value becomes at least 1,120 (1x threshold of 1,120), 2,240 (2x threshold) and 3,360 (3x threshold) respectively at these time units (marked in bold in the $\sum v_i$ row). The corresponding cash inflows are displayed in the PBPP row in figure 4.2. The NPV of the cash inflows is 2,760.96 and the total project NPV is 1,513.75.

- EBPP: The benchmark cost B for the project equals 1,470 (= $\frac{140+170}{2} + \frac{100+200}{2} + \frac{280+360}{2} + \frac{240+280}{2} + \frac{200+240}{2} + \frac{330+400}{2}$) and the threshold based on the costs is set to 490 (= 1,470/3). Payments are received at times 7 (1,593 = 0.9 x (300 + 360 + 660 + 900 x 0.5)) and 12 (810 = 0.9 x (300 + 360 + 660 + 900 + 540) - 1,593), since at these time units the total costs becomes at least 490 and 980 respectively. The third payment is received at the project deadline and equals 621 (= 0.9 x 3,360 - 1,593 - 810). The time instances for which the values for $\sum c_{im}$ exceed the threshold are marked in bold in figure 4.2. The cash inflows, which are received at these time instances, are displayed in the EBPP row in figure 4.2. The NPV of the cash inflows is 2,743.58 and the total project NPV is 1,496.37.

- TBPP: Payments are received at times 5, 10 and 14 since $\lceil \delta_{n+1}/K \rceil$ equals 5. The cash inflows received are 783 (= 0.9 x (300 + 360 x 0.67 + 660 x 0.5)), 1,506.60 (= 0.9 x (300 + 360 + 660 + 900 + 540 x 0.6) - 783) and 734.40 (= 0.9 x 3,360 - 783 - 1,506.60) respectively (displayed in the TBPP row of figure 4.2), which amount to a NPV of 2,746.50. The total project NPV is 1,499.28.

4.3 Solution representation & schedule generation

To properly represent a solution of the DTCTP–NPV, we employ two types of lists. The first is a mode list (ML), which contains the number of the selected mode for each activity in ascending order of the activity numbers, i.e. the first position in the ML holds the selected mode for the first activity, and the last position contains the value for the last activity. For the second list, a solution encoding is required to determine activity finish times. The activity list (AL) and random key (RK) representations are most commonly used (Kolisch and Hartmann, 1999). In the AL representation, the position of an activity in the list determines its relative priority, whereas in the RK representation, the priority value attributed to each activity determines the order of scheduling. We use two alternatives, namely a finish time list (FTL) and a slack list (SL). We discuss the ML, FTL and SL in the following sections.



$\sum v_i$	0	100	200	300	585	870	**1,155**	1,770	**2,328**	2,436	2,544	2,652	2,760	3,060	**3,360**
$\sum c_{im}$	0	0	0	140	140	140	240	**520**	800	800	800	800	**1,040**	1,040	**1,370**
PBPP	0	0	0	0	0	0	**1,040**	0	**1,056**	0	0	0	0	0	**929**
EBPP	0	0	0	0	0	0	0	**1,485**	0	0	0	0	**810**	0	**621**
TBPP	0	0	0	0	0	**783**	0	0	0	0	**1,507**	0	0	0	**734**

Threshold: 1,120

Threshold: 433

Figure 4.2: Example data & schedule.

Figure 4.3 displays a flowchart of the proposed schedule generation, and distinguishes between the FTL and SL representations and their corresponding approaches. The schedule generation is employed to evaluate the solutions of the metaheuristics in section 4.4.

4.3.1 Mode list and deadline–feasibility

Since the selected modes for an activity determine both its duration and its cost, it is possible that the project's earliest finish time exceeds the deadline. In such a case, we apply a deadline feasibility improvement method, which changes activity modes until a deadline–feasible combination is found. The method changes the mode of a random activity on the critical path to the least expensive mode in terms of costs with a lower duration. Afterwards, the critical path and minimum project duration are recalculated. If the ML is now deadline feasible, the procedure terminates. Otherwise, it is repeated until a feasible mode combination has been found. These steps correspond with $ef_{n+1} \leq \delta_{n+1}$ and *Mode change* in figure 4.3.

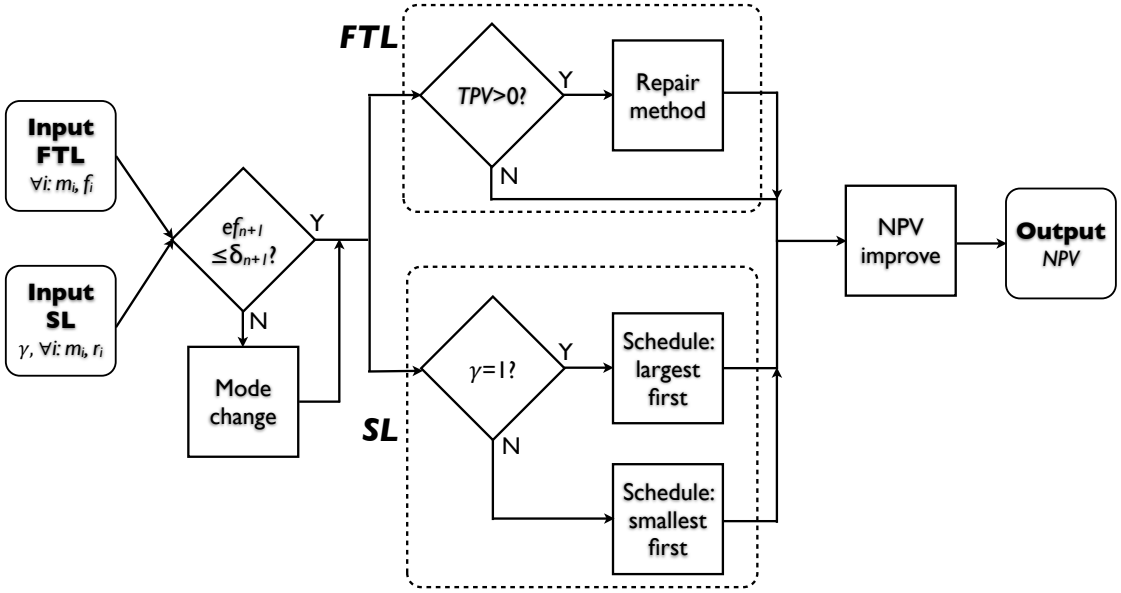


Figure 4.3: Flowchart schedule generation.

4.3.2 Finish time list

In this section, we discuss the scheduling approach which corresponds with the FTL part of figure 4.3. A finish time list (FTL) holds the finish time of each activity, with the possible values ranging from the activity's earliest finish time to its latest finish time. In order to schedule a FTL provided by a metaheuristic, it is sufficient to simply introduce the finish times from the list in the project schedule. Whereas such a schedule is always feasible with respect to the project deadline, a major drawback is, however, that the precedence relations may be violated, i.e. a predecessor ends later than the start time of its successor. The total predecessor violation (TPV) is used to denote the total precedence violation and holds the total number of time units of overlap between activities: $\sum_{i=1}^{n} \sum_{j=1}^{|S_i|} max(0, f_j - d_{jm} - f_i)$. To remedy these overlaps, we propose two variants of a repair method.

Both variants consist of a forward and backward step. In the first variant (*Repair1*), the forward step moves all activities as late as possible, given the project deadline and the precedence relations. This step considers the activities in decreasing order of finish time, starting from the activity with the highest finish time. Subsequently, the backward step advances all activities as early as possible, in order of increasing start time. Algorithm 5 displays the resulting repair method.

A major downside of the first variant, however, is that the obtained presence feasible finish times may deviate considerably from the FTL provided by the metaheuristic. More specifically, activities which need not have been moved to improve the feasibility, may also

Algorithm 5 Precedence feasibility improvement 1

Repair1 ()
  Sort activities in decreasing order based on their finish time: $List[]$
  For $k = 1$ **to** $|N|$: $i = List[k]$, $max = \delta_{n+1} - f_i$
    For $j \in S_i$
      If $f_j - d_{jm} - f_i < max$ **then** $max = f_j - d_{jm} - f_i$
    End for
    $f_i = f_i + max$
  End for
  Sort activities in increasing order based on their start time: $List[]$
  For $k = 1$ **to** $|N|$: $i = List[k]$, $max = f_i - d_{im}$
    For $j \in P_i$
      If $f_i - d_{im} - f_j < max$ **then** $max = f_i - d_{im} - f_j$
    End for
    $f_i = f_i - max$
  End for

have been assigned a different finish time. Hence, we propose an adjusted version of this repair method as well (*Repair2*). The second variant considers the cumulative successors and predecessors of an activity i (CS_i and CP_i respectively). In the forward step, only activities which overlap with one of their predecessors, or for which at least one of the cumulative predecessors has an overlap, are delayed. Similarly, the backward step only advances activities for which an overlap exists, or for which at least one of the cumulative successors has an overlap. Finally, now that we have obtained a precedence feasible schedule, we aim to minimize the deviation between the original finish times before the repair method and the ones afterwards. Starting from the activity with the highest finish time, we evaluate the deviation between both finish times, and if the deviation is positive, we delay the activity by the minimum of the deviation and the allowable delay based on immediate successors. The pseudocode of the second variant is shown in algorithm 6, with $countCP_i$ ($countCS_i$) the number of cumulative predecessors (successors) of activity i which violate precedence constraints (including i), and bf_i the back–up or original finish time of activity i before the repair method.

4.3.3 Slack list

In this section, we discuss the scheduling approach which corresponds with the SL part of figure 4.3. A slack list (SL) holds a value r_i, with $r_i \in [0; 1[$, for each activity i and displays the percentage of the available slack s_i which should be used by an activity. The finish time of activity i is set as $ef_i + \lfloor (s_i + 1) \cdot r_i \rfloor$. As a result, the finish for an activity always lies between its earliest and latest finish time. Let us illustrate the use of the SL with an example. Assume that we aim to schedule a project of three activities and the selected ML is (1, 3, 2) and the SL is (0.42, 0.83, 0.57). Assume, furthermore, that

Algorithm 6 Precedence feasibility improvement 2

Repair2 ()
  Sort activities in decreasing order based on their finish time: $List[]$
  For $k = 1$ **to** $|N|$: $i = List[k]$, $max = \delta_{n+1} - f_i$
    If $countCP_i > 0$
      For $j \in S_i$
        If $f_j - d_{jm} - f_i < max$ **then** $max = f_j - d_{jm} - f_i$
      End for
    End if
    Else $max = 0$
    $f_i = f_i + max$
  End for
  Sort activities in increasing order based on their start time: $List[]$
  For $k = 1$ **to** $|N|$: $i = List[k]$, $max = f_i - d_{im}$
    If $countCS_i > 0$
      For $j \in P_i$
        If $f_i - d_{im} - f_j < max$ **then** $max = f_i - d_{im} - f_j$
      End for
    End if
    Else $max = 0$
    $f_i = f_i - max$
  End for
  Sort activities in decreasing order based on their finish time: $List[]$
  For $k = 1$ **to** $|N|$: $i = List[k]$
    If $bf_i > f_i$ **then** $max = bf_i - f_i$
      For $j \in S_i$
        If $f_j - d_{jm} - f_i < max$ **then** $max = f_j - d_{jm} - f_i$
      End for
      $f_i = f_i + max$
    End if
  End for

activities 1 and 2 have already been scheduled and that based on their finish times and the ML it is concluded that activity 3 has an earliest finish time of 7 and has a slack of 3. This implies that activity 3 has 4 possible finish times, namely 7, 8, 9 and 10. The finish time of activity 3 can be calculated as follows: $f_3 = 7 + \lfloor (3+1) \cdot 0.57 \rfloor = 9$.

Since the available slack of an activity, however, depends on the finish times of other activities which have already been scheduled, the activity slack needs to be updated every time an activity is scheduled. For scheduling the activities we use algorithm 7. The activity slack is dynamically calculated, based on the early and latest finish times of other activities and takes the selected activity modes m into account. The dynamic slack of activities which have been scheduled is set to zero, whereas the slack of the unscheduled activities is reevaluated each time an activity is scheduled. U is the set of unscheduled activities and F the set of scheduled activities. Based on a binary directional variable γ, we first schedule the activities with the largest SL value r_i ($\gamma = 1$) or the activities with the smallest r_i ($\gamma = 0$). The value for γ is determined by the employed metaheuristic (section 4.4). An advantage of working with the proposed scheduler is that, unlike for

the FTL representation, no precedence infeasible finish times can be assigned, due to the recalculation of the activity slack.

Algorithm 7 Schedule generator slack list

SchedGenSL (binary directional variable γ)
Set $U = \{1, 2, \ldots, n\}$; $F = \emptyset$; $count = |N|$; $ef_0 = lf_0 = 0$; $ef_{n+1} = lf_{n+1} = \delta_{n+1}$
While $count > 0$
For $i = 1$ **to** $|N|$
If $i \notin F$ **then** calculate $ef_i = max\{ef_j + d_{im} | j \in P_i\}$
End for
For $i = |N|$ **to** 1
If $i \notin F$ **then** calculate $lf_i = min\{lf_j - d_{jm} | j \in S_i\}$
End for
For $i = 1$ **to** $|N|$ **do** $s_i = lf_i - ef_i$
If $\gamma = 1$ **then** find k: $r_k = max\{r_j | j \in U\}$
Else find k: $r_k = min\{r_j | j \in U\}$
$f_k = ef_k + \lfloor (s_k + 1) \cdot r_k \rfloor$; $ef_k = lf_k = f_k$; count=count-1; $U = U \setminus \{k\}$; $F = F \cup \{k\}$
End while

It can be observed that the proposed SL representation is somewhat similar to the float factor proposed by Tavares et al. (1998) to analyze project risk, since both allow for scheduling an activity between its earliest and its latest finish time. The major difference is, however, that the float factor is assumed to be the same for all activities, whereas the SL representation does not necessarily employ the same r_i for each activity.

4.3.4 NPV improvement

Once the project has been scheduled, either based on the FTL or on the SL representation, we improve the project NPV in the following manner. Starting from the activity i with the largest finish time f_i, the NPV improvement checks whether the activity i is scheduled earlier than its latest finish time lf_i, and if activity i is scheduled entirely within a payment interval. The latter condition implies that f_i is smaller than a time instance t for which $y_t = 1$, and that the start time of activity i is greater than or equal to the previous w ($w < t$) for which $y_w = 1$: $f_i \in [max\{w | y_w = 1 \wedge w < t\}, t | y_t = 1[$. If both conditions are met, the activity i is delayed as much as possible in the payment interval: $f_i = min\{min\{f_j - d_{jm} | j \in S_i\}, t\}$. Otherwise, we continue with the next activity with the largest finish time. The procedure terminates once the activity with the smallest finish time has been reached and evaluated. The updated project NPV is subsequently calculated. The NPV improvement method corresponds with *NPV improve* in figure 4.3.

As an example, assume an activity i with a start time of 12 and a finish time of 14, and two payments at times 11 and 19. Since the activity's current start time 12 is larger than the previous payment time of 11 ($f_j - d_{jm_j} \geq w$) and its current finish time 14 is smaller than the next payment time of 19 ($f_i < t$), the NPV improvement method delays

the activity by 5 time units such that its finish time now equals 19. This way, the NPV of the activity's cash outflows is reduced without lowering the NPV of the cash inflows. Alternatively, assume the activity has a successor with a start time of 17. In this case, the activity is only delayed by 3 time units and its new finish time is 17 ($=min\{17, 19\}$).

More complex NPV improvement methods (chapters 2 & 3) are not employed in this chapter, since the goal of the research here is to investigate the impact of different (atypical) solution representations, rather than propose a new scheduling technique. Furthermore, the problems discussed (section 4.2) consider both timing and size of payments, unlike previous chapters. As a result, there is an interaction between the timing and size of cash inflows, which may result in delays of activities having the opposite effect on project NPV than what was desired. Simply put, if both timing and size of cash flows are variable, delaying an activity beyond a payment time (e.g. based on the scheduler from chapter 3), may delay the payment time as well, reducing the NPV instead of increasing it.

4.4 Genetic algorithm

In this section, we provide a detailed overview of our solution method for the DTCTP–NPV with the three payment models, based on a genetic algorithm (GA).

A GA was first proposed by Holland (1975) and employs aspects from evolutionary biology, namely selection, crossover and mutation operators. The goal is to combine existing individuals into new solutions in order to improve the algorithm's objective. A selection operator chooses individuals, i.e. parents, from the population and applies a crossover operator to generate better individuals, i.e. children. The crossover part constitutes the intensification step of the GA whereas the diversification step is applied as a part of mutation operator. This operator makes adjustments on a small percentage of population elements to ensure that diverse solutions exist in the population. Afterwards, a population update is done to retain the best individuals and omit the worst solutions. The selection, crossover and mutation steps are repeated until a stopping criterion has been reached.

Figure 4.4 displays an overview of the GA implementation. We use the notations P_0 and P_1 to clearly distinguish between the current population of elements and the newly generated children. It is assumed that both populations have the same size. Starting from an initial population, new elements are generated (intensification), after which a diversification step is applied to ensure that the procedure does not get stuck in local optima. These intensification (crossover) and diversification (mutation) steps are repeated until a stop criterion has been reached. Both the FTL and the SL representation can be included in the GA. In the following subsections we go into detail about each of the procedures' steps.

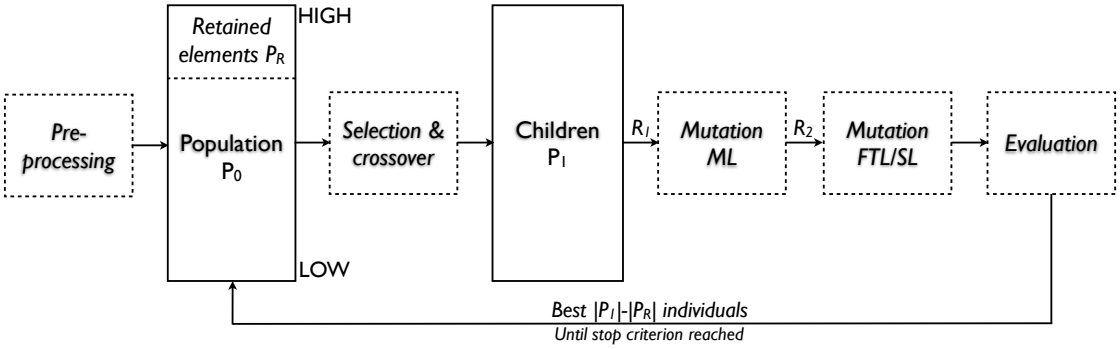


Figure 4.4: Flowchart genetic algorithm.

4.4.1 Preprocessing

Before we start with the GA metaheuristic, a preprocessing step is required in order to eliminate infeasible modes. The proposed preprocessing method is the elimination of long modes procedure of Akkan et al. (2005). An infeasible mode can be described as a mode which always leads to a deadline violation if selected. This can be tested for every mode of an activity by selecting the shortest mode for all other activities and calculating the minimum project duration. If this duration exceeds the project deadline, the selected mode is infeasible and should not be taken into further consideration. For each activity we start with the shortest mode such that, if an infeasible mode is found, any modes of the same activity with a longer duration can also be immediately omitted.

4.4.1.1 Initial population

In the initial population we randomly generate $|P_0|$ times both a ML and a FTL/SL, with P_0 the population of the GA. The FTL is generated in between each activity's earliest and latest finish time. For this representation, we explicitly ensure that at least one precedence feasible element is included, by setting the finish time f_i of each activity equal to its corresponding earliest finish time ef_i for this single solution. A SL value is chosen from the interval $[0; 1[$. The binary direction variable γ of section 4.3.3 is used to determine how the project schedule should be constructed by algorithm 7 for the SL representation, and is randomly set to 0 or 1.

All elements are subsequently evaluated, based on the schedule generation of figure 4.3. The best $|P_R|$ elements are determined and are considered the elite individuals of the population.

4.4.1.2 Selection

Since a GA typically combines existing solutions to create new solutions, a selection operator is required to determine what individuals are used for crossover. In the proposed GA, we employ the elite selection operator of chapter 2 (Leyman and Vanhoucke, 2015). This operator selects the father from the set of best elements P_R, and the mother based on a four–tournament selection from the entire population P_0.

4.4.1.3 Crossover

A crossover operator in a GA combines several existing individuals (parents) into new individuals (children), with the goal of generating better solutions (intensification). We propose to always combine two parents and always produce two children. To do so, we apply a one–point crossover to both the ML and the FTL/SL, with the same cut–off point. We continue generating children until we have a total of $|P_1|$ $(=|P_0|)$ elements. For the SL, the directional variable γ is randomly copied from one of the parents.

4.4.1.4 Mutation

To diversify the newly generated children, a mutation operator is used. For the ML, we apply a random mode change on each activity with a probability of R_1. The new mode is required to be different from the old mode. Once the mutation operator has finished the mode change, we evaluate the deadline feasibility of the ML. If the adjusted ML is deadline infeasible, we employ the deadline feasibility improvement method of section 4.3.1. If the FTL representation is used, we have to additionally check whether the FTL values lie within their feasible range based on the available slack in the altered ML. If this is not the case for an activity, it is assigned a random FTL value in between its updated earliest and latest finish time.

In terms of mutation of the FTL, we randomly select a different value between the activity's earliest and latest finish time. In the SL representation, a different value from the interval $[0;1[$ is chosen. In both cases, a mutation probability R_2 is used for each activity.

4.4.1.5 Population evaluation & update

After the different genetic operators have been applied, we apply the scheduler of figure 4.3 for each individual of P_1. We subsequently introduce the best $|P_1| - |P_R|$ children in the population P_0. The best $|P_R|$ parents are retained. Afterwards, we reevaluate which $|P_R|$ elements are the best in the updated population.

4.5 Computational results

In this section, we discuss our computational results for the DTCTP–NPV with the three payment models. We provide details of the data used and configure our algorithms. Our results are compared with the work of He et al. (2009b). We employ the 5,000 schedules stopping criterion as defined by Lova et al. (2009) for all tests, and the discount rate is set to 1%.

4.5.1 Test data

The problem set used in our experiments is the dataset of Demeulemeester et al. (1998), which contains 1,800 AoN instances for the DTCTP. We have generated a created value v_i for each activity from the interval [101;150], to ensure that an activity's created value is always larger than the cost of any of its modes (table 4.2). The benchmark cost B for the EBPP is calculated as the sum of the average mode costs: $B = \sum_{i=1}^{|N|}(\sum_{m=1}^{|M_i|} c_{im}/|M_i|)$. For the TBPP, payments are assumed to occur every 10 time units.

We extend the data with a deadline, which varies from the shortest to the longest project duration in steps of 0.25, based on a parameter D. Additionally, we include a parameter K for the number of payment times, which is set to 4, 6 or 8, and a parameter θ for the compensation proportion. The value for θ is 0.7, 0.8 or 0.9. In total, 81,000 (= 1,800 x 5 x 3 x 3) test instances are used for each payment model. Both the instances of Demeulemeester et al. (1998) and the file with created values, are available online at *www.projectmanagement.ugent.be*.

An overview of the data parameters is provided in table 4.2. A comparison is made with the data parameters used by He et al. (2009b). It can be concluded that our proposed test design allows for more variation in the data parameters. Additionally, the data of He et al. (2009b) is not publicly available, whereas our data is available online.

Parameter	**Values this chapter**	**Values He et al. (2009b)**
Number of activities (*Act*)	10, 20, 30, 40 or 50	10, 20, 30 or 40
Number of modes (*Modes*)	2, 4 or 6 or from [1,3], [1,7] or [1,11]	2
Coefficient of network complexity (*CNC*)	1.5, 1.8 or 2.1	NA
Size of durations and costs (*Size*)	Both from [1,20] or [1,100]	d_{im}: [1,10], c_{im}: [10,24]
Deadline increase factor (D)	0, 0.25, 0.50, 0.75 or 1	From [0,1]
Number of payments (K)	4, 6 or 8	3, 4 or 5
Compensation proportion (θ)	0.7, 0.8 or 0.9	0.75, 0.85 or 0.95
Total number of instances	81,000	4,320

Table 4.2: Parameter settings of test instances.

4.5.2 Algorithm configuration

We go into detail about the two alternatives proposed, namely a GA with either a FTL or SL representation (GA–FTL & GA–SL). All tests are run on 20% of the data. An overview of the best found parameter values for each algorithm is shown in table 4.3.

	$\|P_0\|(=\|P_1\|)$	$\|P_R\|$	R_1	R_2
GA–FTL	100	5	0.05	0.03
GA–SL	20	4	0.03	0.05

Table 4.3: Algorithm parameters.

Both repair methods of section 4.3.2 are compared based on their average NPV (*AvNPV*) and based on the percentage average difference between both (*%AvDiff*) in table 4.4. The p–values are those of a paired samples t–test. The results in table 4.4 confirm our earlier assumption that *Repair1* is less efficient than *Repair2* due to unnecessary moves.

	Repair1	**Repair2**	**Difference**	
	AvNPV	*AvNPV*	*%AvDiff*	*p–value*
PBPP	1,115.38	1,190.05	6.69	<0.001
EBPP	1,088.13	1,121.52	3.07	<0.001
TBPP	1,081.20	1,109.91	2.06	<0.001

Table 4.4: Comparison of two repair methods FTL.

Table 4.5 displays the results for both representations based on *AvNPV* for the three payment models, and based on *%AvDiff*. The column p–value reports the corresponding p–value of a paired samples t–test, which compares both options. It can be concluded that the results for the SL representation are best for all three payment models. Hence, we can safely assume that the SL representation is a more effective solution representation for the DTCTP–NPV than the FTL.

	GA–SL	**GA–FTL**	**Difference**	
	AvNPV	*AvNPV*	*%AvDiff*	*p–value*
PBPP	1,190.05	1,215.54	2.14	<0.001
EBPP	1,121.52	1,206.93	7.62	<0.001
TBPP	1,109.91	1,146.45	3.29	<0.001

Table 4.5: Comparison of two representations.

In table 4.6 we compare the results of the GA–SL algorithm with (*Full*) and without the NPV improvement (*NoImpr*) of section 4.3.4. The results show an added value of the

NPV improvement (all three p–values <0.001), but it can be noted that the difference is on average rather small. This rather small improvement is due to the NPV improvement only delaying activities under very specific conditions (activity entirely within a payment interval).

	NoImpr	**Full**	**Difference**	
	AvNPV	*AvNPV*	*%AvDiff*	*p–value*
PBPP	1,213.64	1,215.54	0.16	<0.001
EBPP	1,203.33	1,206.93	0.30	<0.001
TBPP	1,145.22	1,146.45	0.11	<0.001

Table 4.6: Added value of NPV improvement.

In figure 4.5 the convergence of the GA–SL for the PBPP is displayed, with the average number of generations of the GA on the horizontal axis and the average NPV on the vertical axis. Since the final values in the graph, i.e. those with an approximately 240 generations, correspond with 5,000 schedules, it can be concluded that the GA–SL has a good convergence. In the first couple of generations the NPV improves considerably, but the improvement becomes progressively less as more generations are run. Similar graphs can be constructed for the EBPP and TBPP.

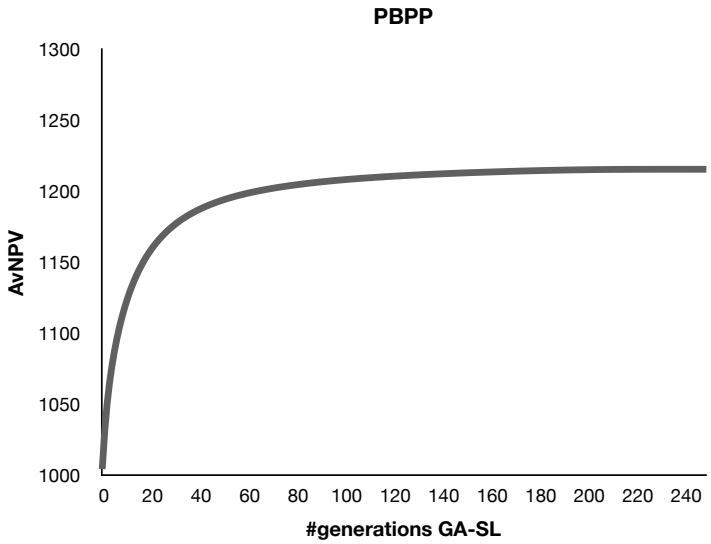


Figure 4.5: Convergence GA–SL algorithm (PBPP).

4.5.3 Comparison with literature

We compare the results of our GA–SL with the results of the simulated annealing (SA) algorithm of He et al. (2009b). The major differences between their work and ours, can be summarized as follows:

- He et al. (2009b) claim to solve a multi-mode payment scheduling problem. Given existing work on multi-mode project scheduling (see e.g. the overview paper of Hartmann and Briskorn, 2010), we claim that a multi-mode problem in general requires renewable resources in the problem definition. Time/cost trade-offs, which constitute the actual problem discussed by He et al. (2009b), are a subclass of multi-mode problems without renewable resources and only one resource, i.e. a single non–renewable resource. Furthermore, the authors limit their data parameters to a point where we believe the question can be raised whether He et al. (2009b) not simply discuss a simplified variant of a DTCTP, i.e. only two modes per activity.

- He et al. (2009b) assume the AoA network representation. As a result, payments of cash inflows can only occur upon event completion, which always correspond with the finish time of one or more activities. We, however, employ an AoN representation, which results in payments able to occur at any time instance between the start and end of the project (end included). This way, the problem discussed in this chapter is in fact more complex than the one used by He et al. (2009b), but also more realistic and more flexible in terms of occurrence of payments.

Based on these differences, we believe that the algorithm of He et al. (2009b) can be used for comparison, but the results should be taken with a pinch of salt. We have implemented the SA ourselves since the data used by the authors is not publicly available. The following changes have been made to the SA algorithm, in order to improve performance:

- The preprocessing method of section 4.4.1 is included, to allow for the same mode elimination as in our approach.

- The deadline feasibility improvement method of section 4.3.1 is inserted to improve the ML feasibility of newly generated lists. The SA discussed in He et al. (2009b) on the contrary continues searching by randomly generating a new neighbor of the current ML, until a feasible mode combination has been found.

- The activity finish times are updated after a mode change if any precedence relations are violated. We delay successors or advance predecessors to ensure precedence feasibility. If no feasible schedule can be constructed, another neighbor of the ML is generated, until a feasible list has been found.

- We include the 5,000 schedules criterion as stopping condition for the outer loop of the SA, to ensure a proper comparison with our proposed solution method.

Table 4.7 provides an overview of the comparison of our GA–SL approach with the SA of He et al. (2009b), based on the average project NPV reported for both algorithms. The column $\%\Delta$ reports the percentage difference between both approaches. The statistical significance of the difference between the two algorithms is tested based on a paired samples t–test and reported a p–value < 0.001. Based on the results in the table, the following can be concluded:

- Overall, the GA–SL outperforms the SA for all three payment models. The largest improvement is found for the EBPP model.
- The relative performance of the GA–SL compared to the SA increases for larger values of *Act* and *Modes*, which implies that our approach scales well as the problem complexity increases.
- Larger deadlines lead to an improved performance of the GA–SL, signifying that this algorithm is better at handling larger solution spaces.
- Increases in the number of payments K increase project NPV, but decrease the difference between both algorithms, in particular for the PBPP and EBPP.

In table 4.8, we compare the average computation times of the GA–SL and SA approaches. It can be concluded that the GA–SL algorithm leads to considerably smaller computation times than the SA algorithm. The effect is found to be particularly strong for increases in the data parameters *Act*, *Size* and D, as shown in the table. Overall, the EBPP is slowest and the PBPP and TBPP are fastest, for the SA and GA–SL method respectively.

Based on the results of the GA–SL approach, several observations can be made:

- A larger average project NPV is obtained for the PBPP and EBPP compared to the TBPP (table 4.7). The PBPP and EBPP allow for a greater degree of flexibility by the contractor, since the payment times are linked to the project schedule. This way, the contractor can ensure payments are received earlier by effectively optimizing the receipt of cash inflows. In the TBPP on the contrary, the payment times are fixed which implies less flexibility for the contractor. Thus, the contractor can increase his NPV by obtaining variable payment times.
- The top left graph of figure 4.6 displays the impact of the *CNC* (network structure) on the project NPV. These results are in line with literature (e.g. Herroelen et al.,

		PBPP			EBPP			TBPP		
		GA–SL	SA	%Δ	GA–SL	SA	%Δ	GA–SL	SA	%Δ
Act	10	507.46	485.73	4.47	494.72	458.76	7.84	497.55	478.41	4.00
	20	896.91	789.20	13.65	886.37	750.05	18.18	856.80	772.83	10.86
	30	1,236.06	1,032.70	19.69	1,225.20	979.05	25.14	1,165.87	1,010.53	15.37
	40	1,571.30	1,254.76	25.23	1,564.03	1,192.14	31.20	1,463.04	1,220.61	19.86
	50	1,865.99	1,489.36	25.29	1,864.32	1,420.02	31.29	1,748.98	1,459.10	19.87
Modes	2	1,150.67	983.70	16.97	1,143.69	938.98	21.80	1,090.41	957.90	13.83
	4	1,246.70	1,026.39	21.46	1,238.91	972.07	27.45	1,174.33	1,004.38	16.92
	6	1,317.14	1,060.57	24.19	1,308.24	1,005.28	30.14	1,225.15	1,038.81	17.94
	[1,3]	1,102.49	947.98	16.30	1,091.72	904.23	20.73	1,052.22	926.74	13.54
	[1,7]	1,224.01	1,103.83	20.73	1,216.16	961.41	26.50	1,155.52	992.38	16.44
	[1,11]	1,252.25	1,029.62	21.62	1,242.85	978.06	27.07	1,181.04	1,009.57	16.98
CNC	1.5	1,198.68	1,005.05	19.27	1,191.30	955.52	24.68	1,132.37	981.14	15.41
	1.8	1,209.82	1,004.28	20.47	1,199.98	954.76	25.68	1,138.68	982.78	15.86
	2.1	1,238.13	1,021.73	21.18	1,229.50	969.74	26.79	1,168.28	1,000.97	16.71
Size	[1,20]	1,975.72	1,741.37	13.46	1,989.73	1,707.47	16.53	1,949.26	1,708.96	14.06
	[1,100]	455.37	279.33	63.02	424.13	212.54	99.55	343.63	267.64	28.39
D	0	1,239.63	1,191.42	4.05	1,219.30	1,158.13	5.28	1,220.49	1,179.28	3.50
	0.25	1,242.31	1,120.73	10.85	1,225.10	1,088.14	12.59	1,203.22	1,102.88	9.10
	0.50	1,216.72	1,017.43	19.59	1,205.97	976.12	23.55	1,151.09	993.24	15.89
	0.75	1,197.07	911.58	31.32	1,195.26	846.47	41.21	1,101.86	884.08	24.63
	1	1,181.99	810.59	45.82	1,189.01	731.16	62.62	1,055.56	782.00	34.98
K	4	1,139.03	933.26	22.05	1,137.50	887.11	28.23	1,064.45	916.41	16.15
	6	1,229.98	1,022.28	20.32	1,220.24	968.99	25.93	1,160.03	997.42	16.30
	8	1,277.62	1,075.51	18.79	1,263.05	1,023.92	23.35	1,214.86	1,051.06	15.58
θ	0.7	1,018.27	856.34	18.91	1,006.54	807.29	24.68	966.88	838.85	15.26
	0.8	1,214.24	1,009.86	20.24	1,206.25	959.42	25.73	1,145.48	988.07	15.93
	0.9	1,414.12	1,164.85	21.40	1,407.99	1,113.30	26.47	1,326.98	1,137.98	16.61
Overall		1,215.54	1,010.35	20.31	1,206.93	960.00	25.72	1,146.45	988.30	16.00

Table 4.7: Comparison with literature: average NPV.

		PBPP		EBPP		TBPP	
		GA–SL	SA	GA–SL	SA	GA–SL	SA
Act	10	0.09	0.19	0.11	0.25	0.08	0.19
	20	0.21	0.64	0.30	0.99	0.20	0.69
	30	0.37	1.46	0.52	2.39	0.34	1.64
	40	0.56	2.61	0.79	4.56	0.50	2.93
	50	0.75	3.63	1.05	7.05	0.67	4.53
Size	[1,20]	0.25	0.72	0.32	1.10	0.23	0.78
	[1,100]	0.55	2.81	0.79	5.00	0.48	3.20
D	0	0.32	1.42	0.38	2.13	0.30	1.47
	0.25	0.36	1.39	0.46	2.16	0.34	1.47
	0.50	0.39	1.66	0.55	2.79	0.36	1.81
	0.75	0.43	1.96	0.65	3.52	0.38	2.24
	1	0.49	2.40	0.74	4.66	0.40	2.98
Overall		0.40	1.77	0.55	3.05	0.36	1.99

Table 4.8: Comparison with literature: computation times (s).

1998) and show that a higher *CNC* value, and hence a higher connectedness of the project network, leads to a better objective function value, i.e. a higher NPV. These

findings are furthermore generalizable to all three models. Hence, more complex networks allow for higher NPV for contractors, because a good schedule is then easier to obtain, since less possible schedules exist.

- As can be observed in the top right graph of figure 4.6, the effect of increases in K on the project NPV is larger from 4 to 6 compared to the increase from 6 to 8. This implies that for relatively few payments a considerable increase in project NPV can be obtained by the contractor, if an increase in the number of payment times can be negotiated with the client. This is particularly true for the TBPP. As such, it is important for the contractor to obtain a sufficiently high number of payment times. In an ideal situation, the contractor would receive payments at every time instance. However, the contractor would then still have to decide on the modes of each activity, e.g. execute an activity in a shorter, more expensive mode, to receive payments earlier or not. Furthermore, a situation such as this assumes that all negotiation power lies with the contractor, in terms of the negotiations with the client to determine the details of the payments' timing and size, which is a rather unrealistic assumption.

- The bottom right graph focusses on the impact of the size of the activity durations and costs (*Size*). The vertical axis displays the proportion in average NPV between the PBPP and EBPP for both a low ([1,20]) and high ([1,100]) value for *Size*. It can be concluded that for low values of *Size* it is better to employ a cost–based reimbursement (EBPP, proportion<100%) and a progress–based reimbursement for higher values (PBPP, proportion>100%). These results indicate that if the costs are relatively low compared to the revenues, the contractor should aim to incur costs early and let the resulting cost curve drive the income of payments, whereas in case of relatively high costs the contractor should delay costs while ensuring project progress based on created value.

4.6 Conclusions & future research

We have discussed three payment models for the deadline variant of the discrete time/-cost trade–off problem with net present value optimization (DTCTP–NPV). A full mathematical model formulation for the three payment patterns has been proposed, based on an activity–on–the–node (AoN) representation. A finish time list (FTL) and slack list (SL) representation have been implemented as part of both a genetic algorithm (GA). The combination of the SL representation and the GA (GA–SL) was shown to perform best.

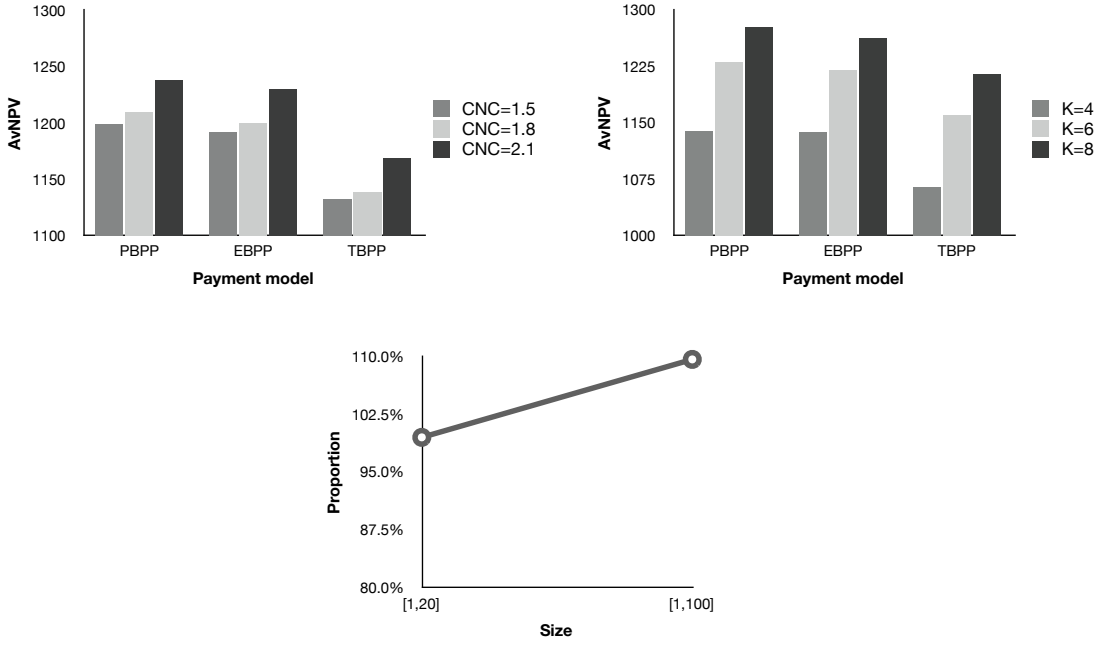


Figure 4.6: Summary insights.

The results of GA–SL have, furthermore, been proven to significantly outperform the best results available in literature. An analysis of the impact of the data parameters showed their effect on the objective function.

Several avenues for future research exist. First, the FTL and SL representations could be compared with the topological ordering (TO) employed in chapters 2 and 3. Second, more complex schedulers, e.g. those of the previous chapters, can be extended or adjusted to determine both timing and size of cash inflows. Third, it may be worthwhile to also consider different models for the cash *outflows* of each activity (see chapter 5), on top of the payment models for cash *inflows*.

5

Capital– and resource–constrained project scheduling with net present value optimization

In this chapter, we study the capital–constrained project scheduling problem with discounted cash flows (CCPSPDC) and the capital– and resource–constrained project scheduling problem with discounted cash flows (CRCPSPDC). The objective of both problems is to maximize the project net present value (NPV), based on three cash flow models. Both problems include capital constraints, which force the project to always have a positive cash balance. Hence, it is crucial to schedule activities in such an order that sufficient capital is available. The contribution of this chapter is threefold. First, we propose three distinct cash flow models, which affect the capital availability during the project. Second, we introduce two new schedulers to improve capital feasibility, one for the CCPSPDC and one for the CRCPSPDC. The schedulers focus on delaying sets of activities, which cause cash outflows to be received at later time instances, in order to reduce capital shortages. Both schedulers are implemented as part of three metaheuristics from literature, in order to compare the metaheuristics' performance. Two penalty functions have been included, one to improve capital feasibility and another to improve deadline feasibility. Third, the proposed procedure has been tested on a large dataset and the added value of the schedulers has been validated. Managerial insights are provided with respect to the impact of key parameters.

5.1 Introduction

In this chapter, we extend the resource–unconstrained max–NPV problem with capital constraints and three cash flow models, and refer to this problem as the capital–constrained project scheduling problem with discounted cash flows (CCPSPDC). Furthermore, the problem is also extended with the presence of renewable resource constraints, and we refer to this problem as the capital– and resource–constrained project scheduling problem with discounted cash flows (CRCPSPDC). In the max–NPV problem and in the RCPSPDC no limit is set on the cash balance (the sum of the cash inflows received and the cash outflows paid) at any particular time, which implies that the cash balance may very well be negative at certain times during the project duration. The CCPSPDC and CRCPSPDC impose the additional constraints that at no point in time the cash balance, or available capital, can be negative. This way, cash outflows can only be paid if sufficient capital is available, whereas cash inflows add to this capital. The cash outflows are included as part of a general model. We present **metaheuristic** solution procedures with new **scheduling techniques** tailored to the needs of the problems, in particular to the capital constraints. The solution procedures in this chapter focus on solving capital shortages by delaying negative cash flows in order to move these cash flows later than the capital shortage.

	Timing	**Size**	**Timing & size**
Cash in	Payments at activities' completion times (PAC)	Progress payments (PP)	Progress based payment pattern (PBPP)
		Payments at event occurences (PEO)	Expense based payment pattern (PBPP)
Cash out	Payments at activities' completion times (PAC)	Resource usage costs	General capital model

Figure 5.1: Overview of the research on project scheduling with NPV optimization in chapter 5.

The remainder of this chapter is organized as follows. A literature overview is given in section 5.2. The mathematical problem definitions of the CCPSPDC and the CRCPSPDC are discussed in section 5.3, whereas two schedulers are the focus of section 5.4. In section 5.5 an overview of the metaheuristics employed is given, whereas the computational experiments and their results are analyzed in section 5.6. We finish with a conclusion and recommendations for future research in section 5.7.

5.2 Literature overview

Figure 5.2 provides an overview of the history of the max–NPV problem, since NPV optimization in project scheduling was first introduced by Russell (1970). The years in the figure indicate when the first research on the problem (extension) was conducted. The distinction is made between three large areas of research with respect to the cash inflows of activities, which depend on the negotiations between the contractor and the client:

- **Timing:** the size of the cash inflows can be determined in advance, but the occurrence of payments depends on the actual project schedule. This is typically the case if cash inflows occur at each activity's completion time. As a result, the contractor can only control the timing of cash inflows, but the size is determined by the client.

- **Size:** the payment times are selected in advance (e.g. progress payments every 10 time units), but the size of the payments can only be determined based on the schedule. The contractor can influence the size of the cash inflows, but not their occurrence times.

- **Both:** the timing and size both depend on the project schedule. In this case, the contractor can determine both, but can only employ a limited number of payments (e.g. Dayanand and Padman, 1997), or the payment times depend on the progress of the project (e.g. based on total costs incurred by the contractor (He et al., 2009a)).

Additionally, two of these research classes, namely the research on timing of cash inflows and on the combination of timing and size, have been extended to include capital and multiple activity modes. In the latter case, each activity can be executed with different time–resource combinations.

For an overview of the project scheduling literature with NPV optimization up to 1997, we refer to Herroelen et al. (1997). A summary of more recent work from 1997 on the max–NPV problem and its extensions is displayed in table 5.1. In the *"Cash in"* columns of the table, we highlight whether the objective includes the determination of the timing and/or size of the cash inflows. The *"Extensions"* columns distinguish the papers in the table based on the inclusion of more specific problem characteristics. A trade–off between multiple modes of an activity (MM) is specified as well. At the bottom of the table, all papers that include capital constraints (C), even those published before 1997, are included since these papers solve a problem closely related to the one of this manuscript.

From the table and figure 5.2 we conclude that the majority of the research on the max–NPV problem has been on the timing of cash inflows and several extensions. Chapters 2 and 3 on the single– and multi–mode RCPSPDC respectively (Leyman and Vanhoucke,

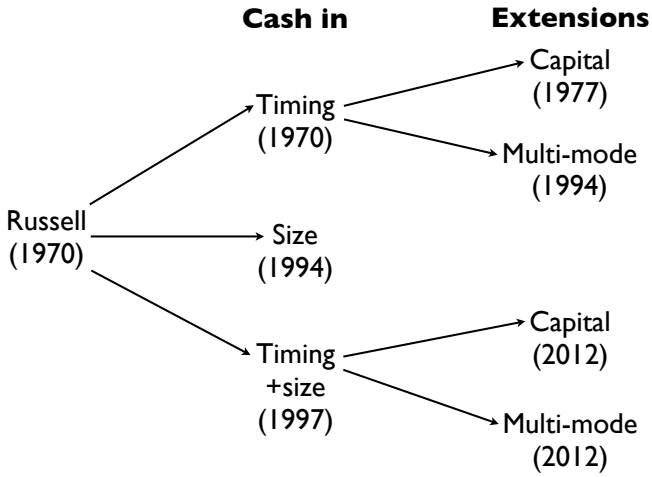


Figure 5.2: History of the max–NPV problem and its extensions.

	Cash in		Extensions	
Authors	**Timing**	**Size**	**MM**	**C**
Dayanand and Padman (1997)	X	X		
Shtub and Etgar (1997)	X			
Etgar and Shtub (1999)	X			
Dayanand and Padman (2001a)	X	X		
Dayanand and Padman (2001b)	X	X		
Schwindt and Zimmermann (2001)	X			
Vanhoucke et al. (2001a)	X			
Vanhoucke et al. (2001b)	X			
Vanhoucke et al. (2003)		X		
Vanhoucke and Debels (2007)	X		X	
He and Xu (2008)	X	X	X	
He et al. (2009a)	X	X	X	
He et al. (2009b)	X	X	X	
He et al. (2014)	X	X	X	
Doersch and Patterson (1977)	X			X
Smith-Daniels and Smith-Daniels (1987)	X			X
Sung and Lim (1994)	X		X	X
Smith-Daniels et al. (1996)	X			X
Özdamar and Dündar (1997)	X		X	X
Özdamar (1998)	X		X	X
He et al. (2012)	X	X	X	X
This work	**X**			**X**

Table 5.1: Literature overview max–NPV problem.

2015, 2016b), however, show that no research has been done on capital restrictions in combination with renewable resource limitations.

In this chapters, we contribute to the literature on NPV maximization in project scheduling in three ways. First, we propose a new scheduling technique as part of a metaheuristic approach for the max–NPV problem with capital constraints, and compare with existing work. Second, we focus on the problem with renewable resource and capital constraints, which has not yet been discussed in literature. We introduce a scheduler which handles both restrictions, while optimizing the project NPV. Third, whereas the focus in literature has been on the timing and/or size of cash *inflows*, we model the timing and size of cash *outflows* as part of a general model. These cash outflows are particularly relevant with capital constraints, since their timing and size have a profound impact on the capital level available during the project (section 5.3).

5.3 Problem definition

In this section, we discuss the problem definitions of the capital–constrained project scheduling problem with discounted cash flows (CCPSPDC), and of its renewable resource–constrained extension the CRCPSPDC.

5.3.1 The capital–constrained project scheduling problem with discounted cash flows

A project can typically be represented by a directed graph or network $G(N, A)$ with N used for the project activities or nodes and A the precedence relations or arcs between the nodes N. We employ the activity–on–the–node (AoN) representation and assume a time–lag of zero for the precedence relations. Each activity i ($i \in N = \{1, \dots n\}$) has a duration d_i, a cash inflow $c_{i,in}$ (> 0) and a cash outflow $c_{i,out}$ (< 0). Additionally, a start dummy 0 and end dummy $n + 1$ are included. The project has a deadline of δ_{n+1}. The finish times f_i of the activities are the decision variables.

Mathematically, the max–NPV problem can be conceptually formulated as follows:

$$\text{Maximize } \sum_{i=1}^{n} (c_{i,in} + c_{i,out}) \cdot e^{-\alpha f_i} \tag{5.1}$$

Subject to:

$$f_i \leq f_j - d_j, \quad \forall (i,j) \in A, \tag{5.2}$$

$$f_{n+1} \leq \delta_{n+1}, \tag{5.3}$$

$$f_i \in int^+, \quad \forall i \in N \tag{5.4}$$

The objective function (5.1) optimizes the project NPV based on a discount rate α,

and discounts both the cash in– and outflow to the activity finish time. Constraints (5.2) enforce precedence feasibility. Constraint (5.3) makes sure the project deadline is met. If no deadline were imposed, cash outflows could be delayed indefinitely. Finally, constraints (5.4) state that the decision variables should be integer values.

In objective function (5.1), it is assumed that cash in– and outflows both occur at activity completion time. However, for the extension with capital constraints (CCPSPDC) we consider alternative occurring times for the cash outflows, since the cash outflows have an impact on the capital balance. We propose the following general model for the CCPSPDC, as an extension to the max–NPV model:

$$\text{Maximize } \sum_{i=1}^{n} c_{i,in} \cdot e^{-\alpha f_i} + \sum_{i=1}^{n} \sum_{t=0}^{d_i} c_{i,out} \cdot v_{it} \cdot e^{-\alpha(f_i - d_i + t)} \tag{5.5}$$

Subject to:

$$C_0 + \sum_{i \in Q_f(t)} c_{i,in} + \sum_{i \in Q_s(t)} \sum_{w=0}^{min(t-(f_i-d_i),d_i)} c_{i,out} \cdot v_{iw} \geq 0, \quad t = 0, \ldots, \delta_{n+1}, \tag{5.6}$$

$$f_i \leq f_j - d_j, \quad \forall (i,j) \in A, \tag{5.7}$$

$$f_{n+1} \leq \delta_{n+1}, \tag{5.8}$$

$$f_i \in int^+, \quad \forall i \in N \tag{5.9}$$

The objective function (5.5) optimizes the project NPV, with the first term containing the NPV of the cash inflows. It is assumed that cash inflows always occur upon activity completion. The second term models the cash outflows based on a parameter v_{it} ($v_{it} \in [0;1]$), which holds the fraction of an activity i's cash outflow to be paid at time $f_i - d_i + t$. In order to ensure that the entire cash outflow $c_{i,out}$ of each activity i is assigned to a time period during the activity's duration, we set $\sum_{t=0}^{d_i} v_{it} = 1$. This way, $c_{i,out}$ is distributed over the activity duration in a predefined manner. Constraints (5.6) model the capital feasibility and ensure that the capital does not become negative at any time unit t. $Q_s(t)$ is the set of activities which have been started on and before time t, and $Q_f(t)$ is the set of activities finished on and before time t. $min(t-(f_i-d_i), d_i)$ in the third sum ensures that for the capital evaluation at time t, only cash outflows of activity i up until time t are considered. Constraints (5.7)–(5.9) are the same as constraints (5.2)–(5.4) in the max–NPV model.

Figure 5.3 provides an overview of five different applications of the general CCPSPDC model. Each separate graph shows the capital profile of a single activity i, with on the horizontal axis the time t and on the vertical axis the capital C_t at time t. The values for C_t are calculated based on constraints (5.6). The bold line in each of the graphs is the

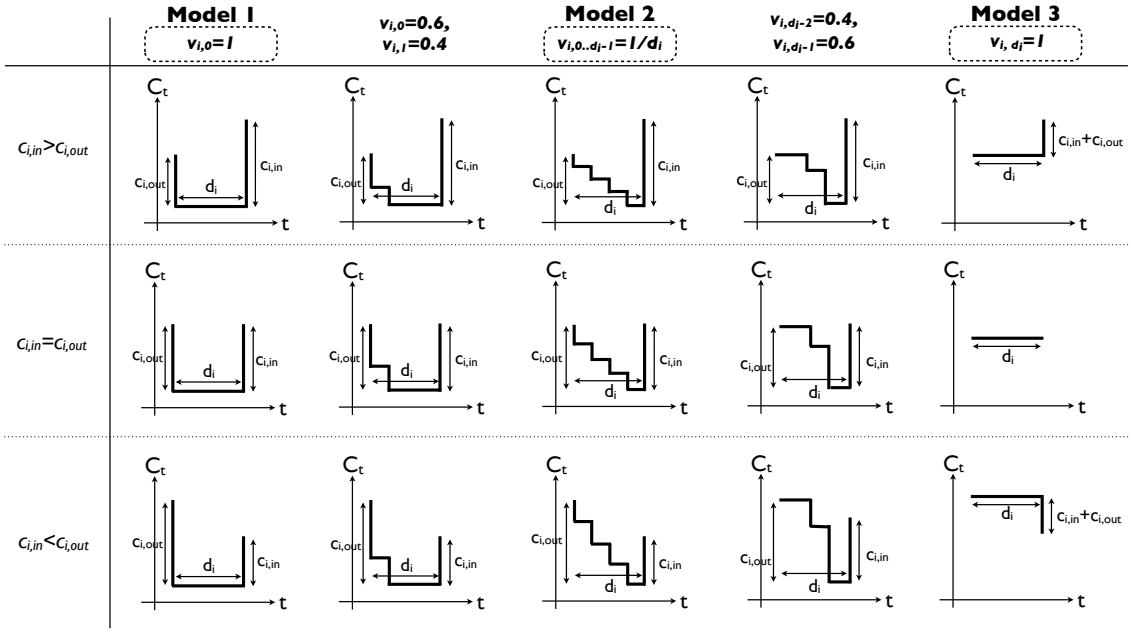


Figure 5.3: Example applications of model CCPSPDC.

capital profile of activity i during its required duration. Each column in the figure displays a different application based on different values for the v_{it} parameters to model the way the cash outflows are spread over the duration, whereas the rows distinguish between the three possibilities regarding the comparative values of $c_{i,in}$ and $c_{i,out}$. In the remainder of this chapter, we focus on the three highlighted applications. Models 1 and 3 constitute two extreme cases, since it is assumed that the cash outflows occur at the start and end of an activity respectively. Without loss of generality, it can be stated that model 2 serves as a representation of all model variations in between both extremes, in which the cash outflows are distributed over the time instances between the activity start and finish times.

- Model 1: cash outflows are paid at activity start times ($v_{i,0} = 1$), and hence reductions in available capital occur at the start of an activity. The resulting objective and capital constraints can be simplified and correspond with functions (5.10) and (5.11) respectively.

$$\text{Maximize} \sum_{i=1}^{n} c_{i,in} \cdot e^{-\alpha f_i} + \sum_{i=1}^{n} c_{i,out} \cdot e^{-\alpha(f_i - d_i)} \tag{5.10}$$

$$C_0 + \sum_{i \in Q_f(t)} c_{i,in} + \sum_{i \in Q_s(t)} c_{i,out} \geq 0, \quad t = 0, \ldots, \delta_{n+1} \tag{5.11}$$

- Model 2: cash outflows are paid on a per time unit basis during the activity duration ($v_{i,0\ldots d_i-1} = 1/d_i$). The objective function and capital limitations are adjusted to functions (5.12) and (5.13) respectively.

$$\text{Maximize} \sum_{i=1}^{n} c_{i,in} \cdot e^{-\alpha f_i} + \sum_{i=1}^{n} \sum_{t=0}^{d_i-1} c_{i,out}/d_i \cdot e^{-\alpha(f_i-d_i+t)} \tag{5.12}$$

$$C_0 + \sum_{i \in Q_f(t)} c_{i,in} + \sum_{i \in Q_s(t)} min(t-(f_i-d_i)+1, d_i) \cdot c_{i,out}/d_i \geq 0, \quad t = 0, \ldots, \delta_{n+1} \tag{5.13}$$

- Model 3: cash outflows are paid at activity finish times ($v_{i,d_i} = 1$), along with the receipt of the cash inflows. The simplified objective function (5.14) and capital constraints (5.15) are shown below.

$$\text{Maximize} \sum_{i=1}^{n} c_{i,net} \cdot e^{-\alpha f_i} \tag{5.14}$$

$$C_0 + \sum_{i \in Q_f(t)} c_{i,net} \geq 0, \quad t = 0, \ldots, \delta_{n+1} \tag{5.15}$$

5.3.2 The capital– and resource–constrained project scheduling problem with discounted cash flows

As an extension to the problem discussed in section 5.3.1, we propose to also include renewable resource (RR) constraints. The mathematical models of section 5.3.1 have to be extended with renewable resource constraints. Each activity i has a resource demand r_{ig} of type g, whereas each RR type g ($g \in R = \{1, \ldots, |R|\}$) has a limited availability of a_g. Functions (5.16) have to be added as additional constraints to the models of section 5.3.1. $S(t)$ are the set of activities in progress at time t.

$$\sum_{i \in S(t)} r_{ig} \leq a_g, \quad \forall g \in R, \; t = 0, \ldots, \delta_{n+1} \tag{5.16}$$

The CRCPSPDC contains both renewable resource and capital constraints, both of which are fundamentally different as discussed along the following lines:

- Renewable resources have a fixed availability, are decreased at the start time of an activity, and are renewed at activity finish time. Examples include machines and workers, which are owned by the company executing the project and do not need to be purchased (Blazewicz et al., 1983; Pritsker et al., 1969).

- Cumulative resources (CR) have a variable availability, which is not necessarily the same after the completion of an activity (Neumann and Schwindt, 2002). Examples of a cumulative resource are inventory and capital.

As an example, consider the network in figure 5.4 with a single RR ($a_1 = 3$), an initial capital C_0 of 30 and a deadline of 10. The discount rate is assumed to be 1%. The graph displays the example's network structure with a duration, RR demand, and cash out– and inflow for each activity. We assume model 1 from section 5.3.1 is applied, with cash outflows at the activity start times and inflows at the activity finish times. We consider the optimal solution to the example and its corresponding schedule in three distinct cases.

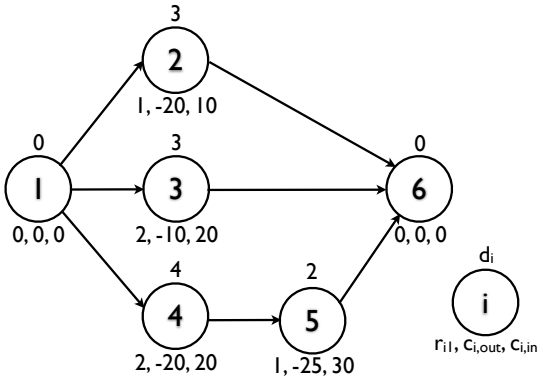


Figure 5.4: Data example.

1. CCPSPDC: the optimal schedule is displayed in the top left of figure 5.5. The width of each of the blocks corresponds with the activities' durations. The bold line displays the capital level available throughout the project (e.g. $C_3 = 30-10-20+20 = 20$). In the schedule, activity 3 is scheduled at its earliest finish time because of its positive NPV, whereas activity 2 is scheduled at its latest finish time due to its negative NPV. The cumulative NPV of activities 4 and 5 is positive so both are scheduled at their earliest finish time, even though the NPV of activity 4 is negative. No capital shortages occur. The project NPV equals 3.26 ($= -18.65 + 9.05 - 10 + 19.41 - 20 + 19.22 - 24.02 + 28.25$). The only trade–off to be made in this case is between the capital availability and the project NPV.

2. RCPSPDC: the optimal schedule is shown in the top right schedule of figure 5.5, with the height of each of the blocks corresponding with its RR demand. The additionally introduced right vertical axis displays the RR availability a_1. Compared to the top right schedule, activities 4 and 5 have been delayed because activities 3 and 4 can

no longer be scheduled in parallel. Since the cumulative NPV of activities 4 and 5 is smaller than the NPV of activity 3, it is best to delay the former activities. It can, however, be observed that the resulting schedule is capital infeasible since the capital drops to -5 between times 7 and 9. The only trade–off in this case is between the RR availability and the project NPV.

3. CRCPSPDC: the optimal schedule can be found at the bottom of figure 5.5. Due to the limited capital availability, activity 2 has to be scheduled at time 7 since otherwise insufficient capital is available to start activity 5. Alternatively, delaying activity 4 and 5 does not ensure a non–negative capital during the project duration. Hence, this case involves trade–offs between capital availability and project NPV, between RR availability and NPV, but also between capital and RR availability.

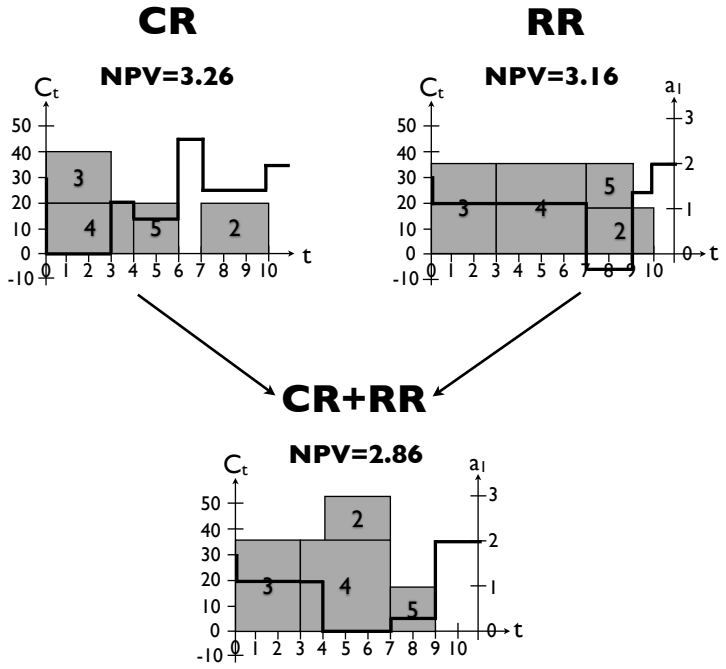


Figure 5.5: Schedules example.

Based on the three different example schedules, it can be concluded that more trade–offs are included in the CRCPSPDC, when compared to both the CCPSPDC and the RCPSPDC.

5.4 Scheduling techniques with capital constraints

In this section, we first discuss our proposed scheduler for the CCPSPDC. Second, we propose a more complex version of the scheduler, which also takes the renewable resource constraints into account. Both schedulers consists of three main parts: an initial schedule, a capital feasibility evaluation or improvement step, and a NPV improvement part.

5.4.1 A scheduler for the CCPSPDC

We start with the construction of an initial schedule and evaluate the capital feasibility. If the schedule is capital feasible (*C–Feas*), a NPV improvement is applied, otherwise a penalty function is employed, used to denote that the resulting schedule is capital infeasible (*C–Infeas*).

The proposed scheduler is applied on a single priority list (PL) generated by a metaheuristic (section 5.5), and is used for each list generated by the metaheuristic. An overview of the overall flow of the scheduler is shown in figure 5.6.

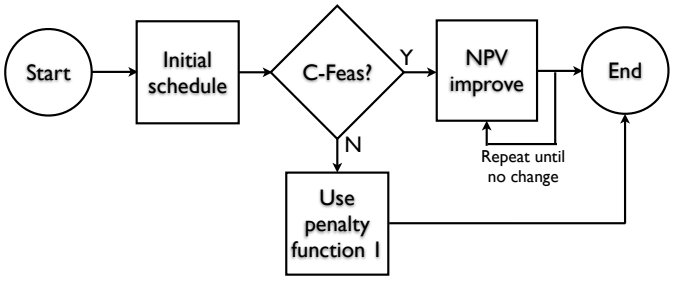


Figure 5.6: Schedule generation flow CCPSPDC.

5.4.1.1 Initial schedule

The initial schedule aims to schedule all activities subject to the precedence relations and to the capital constraints. The activities are scheduled in the order in which they appear in the PL, starting with the first activity in the list. The method aims to find a *C–Feas* finish time and starts from the activity's earliest finish time.

- The partial schedule is *C–Feas*: the finish time of activity i is retained, the capital availability is updated for the entire project duration, and the scheduler continues with the next activity in the PL.
- The partial schedule is *C–Infeas*: the proposed finish time is incremented and the *C–Feas* evaluation is repeated. If the finish time, however, cannot be incremented,

i.e. it would become larger than the latest finish time, no *C–Feas* finish time could be found. Activity i is then scheduled at its earliest finish time and the scheduler continues with the next activity in the PL.

5.4.1.2 Capital feasibility evaluation

Once the initial schedule of section 5.4.1.1 has been completed, we evaluate the capital feasibility of the schedule by calculating the Excess of Capital Request (ECR), which is defined as: $ECR = \sum_{t=0}^{\delta_{n+1}} max(0; -C_t)$, with C_t the capital level at time t. The ECR adds the shortages of capital of all time units t during the planned project duration for which the capital C_t is negative. The resulting ECR value gives an indication of the capital feasibility of the schedule. A large ECR value on the one hand means that many shortages in capital exist or that the existing shortages are large, or both. A small ECR value on the other hand implies that the shortages are limited both in number of periods and in size. Finally, an ECR value of zero means no capital shortages exist.

If the initial schedule of section 5.4.1.1 has an ECR value equal to 0, the scheduler continues with the NPV improvement of section 5.4.1.3. Otherwise, penalty function (5.17) is applied, after which the schedule is returned to the metaheuristic (section 5.5).

$$\begin{cases} NPV = NPV_{C-Infeas} \cdot Y_1^{ECR} & \text{if } NPV_{C-Infeas} \geq 0 \\ NPV = \frac{NPV_{C-Infeas}}{Y_1^{ECR}} & \text{otherwise} \end{cases} \tag{5.17}$$

$NPV_{C-Infeas}$ is the NPV of the *C–Infeas* schedule, whereas Y_1 ($Y_1 \in$[0;1]) is a parameter which is tested in section 5.6.2. Function (5.17) ensures that the NPV of a *C–Infeas* schedule is considerably worse than that of any *C–Feas* schedule by reducing $NPV_{C-Infeas}$, based on the size of the *ECR*. A larger *ECR* value leads to a larger reduction in project NPV.

5.4.1.3 NPV improvement

The NPV improvement is an adjustment of the network–based moves of chapter 2 (Leyman and Vanhoucke, 2015) to improve the project NPV for the RCPSPDC. For a *C–Feas* schedule, the goal is to improve the project NPV while maintaining capital feasibility, by delaying sets of activities with a negative cumulative NPV. These sets are constructed by taking the precedence relations between activities into account. The method of chapter 2 (Leyman and Vanhoucke, 2015) is adjusted by omitting the renewable resources and by including a capital feasibility check of each proposed delay. Once the NPV improvement method is completed, the schedule is returned to the metaheuristic.

5.4.2 A scheduler for the CRCPSPDC

The scheduler for the CCPSPDC is extended to cope with the effects of both the capital and renewable resource limitations. Due to the addition of renewable resource constraints, a more complex scheduler is required. We start with the construction of an initial schedule and continue with the capital feasibility improvement method if the initial schedule is *C–Infeas*. Finally, we go into detail about the changes in activity finish times to improve the project NPV.

The proposed scheduler is applied on a single PL generated by a metaheuristic (section 5.5), and is used for each solution generated by the metaheuristic. An overview of the overall flow of the proposed scheduler is shown in figure 5.7. In general, the scheduler consists of three parts, namely an initial schedule (section 5.4.2.1) and two types of delays:

- The first type of delays (*Capital feasibility improvement*) are discussed in section 5.4.2.2, which aim to improve capital feasibility and delay sets of activities.
- The second type of delays (*NPV improvement*) improve the project NPV by delaying sets of activities with a negative cumulative NPV and are discussed in section 5.4.2.3.

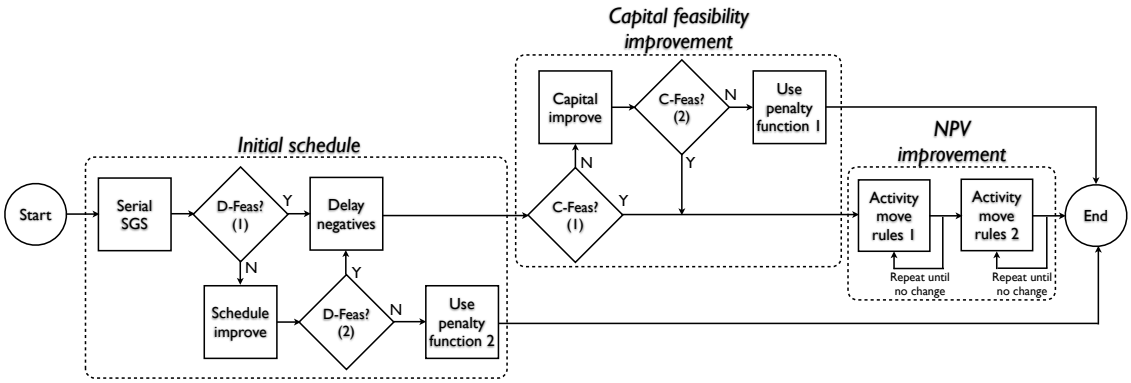


Figure 5.7: Schedule generation flow CRCPSPDC.

5.4.2.1 Initial schedule

For the initial schedule we aim to obtain a deadline feasible (*D–Feas*) earliest start schedule and use the first three steps outlined below. The fourth step improves the earliest start schedule for use in the *Capital feasibility improvement* (section 5.4.2.2) and *NPV improvement* (section 5.4.2.3) methods.

1. The serial schedule generation scheme (SSGS) of Kelley (1963) (*Serial SGS* in figure 5.7) is applied.

2. If the resulting schedule of the first step is deadline–infeasible (*D–Infeas*), the forward–backward improvement method of Li and Willis (1992) is used to reduce the project duration (*D–Feas?(1)* and *Schedule improve* in figure 5.7). This method terminates if no further reductions in project duration can be made. If the schedule of the first step is *D–Feas* the procedure continues with step 4.

3. If the schedule is still infeasible after step 2 the penalty function of chapter 2 (Leyman and Vanhoucke, 2015) is applied (*D–Feas?(2)* check and *Apply penalty function 2* in figure 5.7):

$$\begin{cases} NPV = -Y_2 + NPV_{D-Infeas} \cdot Y_3^{f_{n+1}-\delta_{n+1}} & \text{if } NPV_{D-Infeas} \geq 0 \\ NPV = -Y_2 + \frac{NPV_{D-Infeas}}{Y_3^{f_{n+1}-\delta_{n+1}}} & \text{otherwise} \end{cases} \tag{5.18}$$

   $NPV_{D-Infeas}$ is the NPV of the *D–Infeas* schedule, whereas Y_2 and Y_3 are parameters ($Y_2 > 0$, $Y_3 \in [0;1]$) which are tested in section 5.6.2. Function (5.18) ensures that the NPV of a *D–Infeas* schedule is considerably worse than that of any *D–Feas* schedule by reducing $NPV_{D-Infeas}$ in two ways. First, the parameter Y_3 reduces the project NPV based on the difference between the project duration f_{n+1} and the project deadline δ_{n+1}. Second, Y_2 subtracts a large positive value from $NPV_{D-Infeas}$.

4. All activities i with a negative NPV are delayed as late as possible (*Delay negatives* in figure 5.7). These delays are done in the order in which the activities occur in the solution's PL (section 5.5), starting from the back of the list. If any delays have occurred, the fourth step is repeated until no more changes are applied. The repetition is used to ensure that all activities with a negative NPV are scheduled as late as possible, given the schedule after the previous steps and any delays of other activities.

Once the initial schedule has been generated and the method terminates with step 4, the capital feasibility improvement method of section 5.4.2.2 is applied. If however no *D–Feas* schedule could be found, the method terminates in step 3 and returns the schedule to the metaheuristic, since a *D–Infeas* schedule is not considered for further improvement.

5.4.2.2 Capital feasibility improvement

The goal of the improvement method is to evaluate the capital feasibility of a *D–Feas* schedule generated by the initial scheduler (section 5.4.2.1), and to improve the capital feasibility if the schedule is *C–Infeas*. If the schedule under consideration has an *ECR* equal to zero the algorithm continues with the *NPV improvement* of section 5.4.2.3. If the

ECR is positive, however, a capital feasibility improvement method is applied to reduce the *ECR* value. Figure 5.8 gives an overview of the capital feasibility improvement method discussed in this section, which corresponds with *Capital improve*, *C–Feas?(2)* check, and *Use penalty function 1* in figure 5.7. *C–Feas?(1)* states the condition under which the capital feasibility improvement method is applied.

The goal of the method is to improve a solution's capital feasibility by delaying cash outflows. Delays for activities may require sets of succeeding activities to be delayed as well. These sets can be constructed in two ways, namely based on the network successors and based on the schedule neighbors. In the former case only precedence related activities are included in the set, whereas in the latter case neighboring activities in the project schedule, which may or may not be precedence related, are used. This way, both alternatives consider different sets of activities, which can lead to different changes in the *ECR*. We propose to first apply the schedule–based capital feasibility improvement and second the network–based variant. These delays not only delay sets of activities with a negative cumulative NPV but may also delay sets with a positive cumulative NPV, since the goal is to delay cash outflows to obtain a *C–Feas* solution. We discuss each of the method's large parts (*Check time t*, *Set$_i$ calculations*, *Delay until next positive*, *Delay until next successor* and *Schedule set$_i$*) and the links between the parts in more detail. Table 5.2 provides an overview of the notations used in the remainder of this section.

Check time *t*: the first large part of the capital feasibility improvement method checks the capital feasibility of all time instances t, starting from time 0 to the project deadline δ_{n+1}. If the capital C_t at time t is negative then the set *earlySet* is constructed by including all activities i with a start time smaller than or equal to t in *earlySet*. The capital feasibility improvement method aims to improve the capital feasibility at time t, by delaying the cash outflows corresponding with some or all of these activities i in *earlySet* to time units later than time t. The activities added to *earlySet* are sorted in decreasing order of their appearance in the PL, which implies that the later the position in the PL of an activity i in *earlySet*, the earlier that activity i is considered by the part *Set$_i$ calculations*.

The two possible end nodes in figure 5.8 for the capital feasibility improvement method are linked with the *Check time t* part in the following way. If the project deadline can be reached without any remaining capital shortages, the procedure terminates in the node *End: feas*, and a *C–Feas* schedule has been found. The scheduler continues with the *NPV improvement* part of the algorithm (section 5.4.2.3). If, however, a capital shortage still exists at time t but no more unconsidered activities i remain in *earlySet*, the method finishes in the node *End: infeas*, and no *C–Feas* schedule could be found based on the corresponding PL. In this case penalty function (5.17) is applied, after which the solution

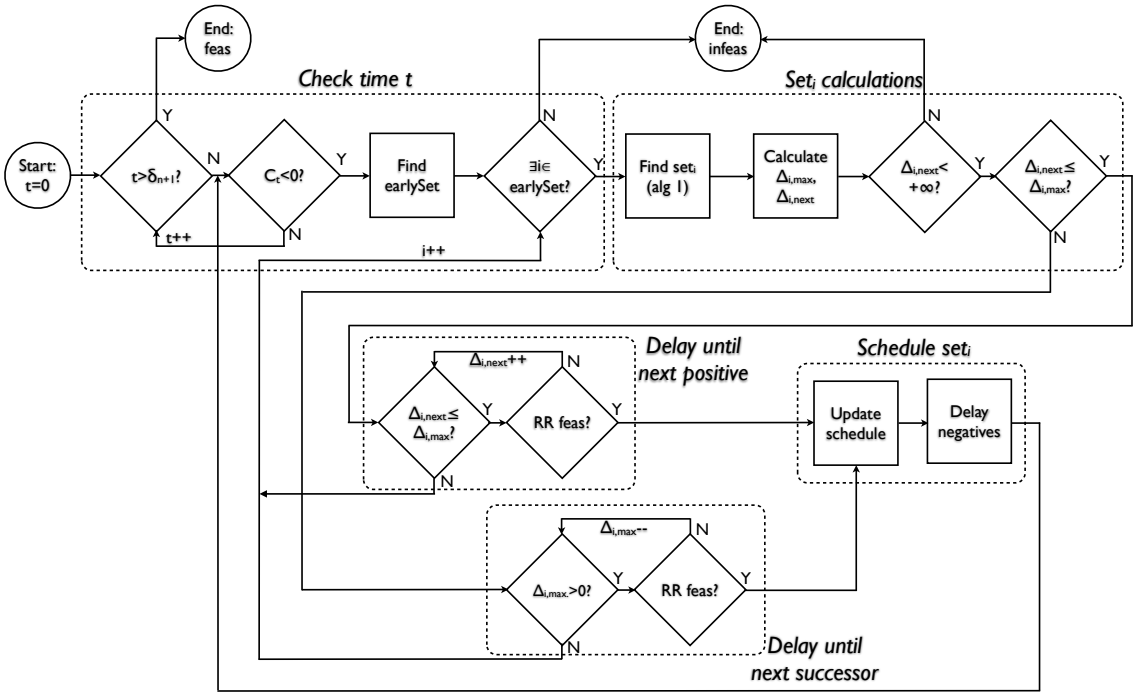


Figure 5.8: Capital feasibility improvement flow.

is returned to the metaheuristic (section 5.5).

Notation	Definition
t	Time at which capital feasibility is evaluated
C_t	Capital available at time t
$earlySet$	Set of activities which may be delayed to improve C_t
i	Index of activity from $earlySet$ currently under consideration
set_i	Set of activities to be delayed together with i based on algorithm 8
k	Index for activities in set_i
S_k	Set of immediate successors of activity k based on the set of arcs A in $G(N, A)$
j	Index of successor of activity k
B_k	Set of neighbors of activity k scheduled after k
h	Index of neighbor of activity k
$\Delta_{i,max}$	Maximum allowable delay for set_i based on any immediate successors not in set_i
l	Index of earliest activity after time t which can reduce the project ECR
$\Delta_{i,next}$	Minimum required delay for set_i based on activity l
$Sched$	Boolean used by algorithm 8 to distinguish between network– and schedule–based variant

Table 5.2: Overview of notations capital feasibility improvement.

***Set$_i$* calculations:** the second part starts with an activity i from $earlySet$ provided by *Check time t* and aims to find the set of activities set_i which has to be delayed together with i, to ensure precedence feasibility. To find set_i, algorithm 8 is applied for the network–

or schedule–based delays:

- Network–based moves ($sched = false$): the algorithm finds the set of all immediate successors of activity i, for which the start time equals the finish time of activity i. The algorithm then recursively determines whether any successors of these successors should also be added.
- Schedule–based moves ($sched = true$): the algorithm finds the set of all neighbors of activity i, for which the start time equals the finish time of activity i. These neighbors are those activities which are scheduled immediately after activity i and which may or may not be precedence related to activity i. The algorithm then recursively determines whether any later scheduled neighbors of these neighbors should also be added.

The capital feasibility improvement method aims to delay set_i to reduce the capital shortage at time t, and calculates two types of delays. The maximum allowable delay for set_i, $\Delta_{i,max}$, based on immediate successors not in set_i is calculated as follows: $\Delta_{i,max} = min(f_j - d_j - f_k | k \in set_i, j \in S_k, j \notin set_i)$. $\Delta_{i,max}$ holds the maximum delay possible for all activities k from set_i, based on their successors j not in set_i.

Algorithm 8 Get all successors

```
GetAllSuc (current activity k,set_i[],boolean sched)
  If sched = false (network–based)
    For j = 1 to |S_k|
      If f_j − d_j = f_k ∧ j ∉ set
        set = set ∪ {j}
        GetAllSuc (j,set_i,sched)
      End if
    End for
  End if
  If sched = true (schedule–based)
    For h = 1 to |B_k|
      If h ∉ set
        set = set ∪ {h}
        GetAllSuc (h,set_i,sched)
      End if
    End for
  End if
  Return set
```

In order to reduce the capital shortage at time t, a second minimum delay has to be calculated. Activity i from *earlySet* has to be delayed at least until the earliest next activity l ($l \notin set_i$), to reduce the impact of activity i on the capital shortage. Such an activity l furthermore has to be scheduled after time t. The goal is to determine a minimum delay for activity i such that its cash outflows are (partially) compensated by

the cash inflow of activity l. Additionally, we do not wish to delay activity i any further than required to reduce the capital shortage at time t, to avoid an unnecessary reduction of the project NPV. The minimum required delay based on such an activity l is called $\Delta_{i,next}$, and depends on the cash flow model used:

- Model 1 (cash outflows occur at activity start time): activity i should be scheduled later than activity l, i.e. $f_i - d_i \geq f_l$, to allow for a proper compensation of $c_{i,out}$.
- Model 3 (cash outflows occur at activity finish time): activity i should have a finish time equal to at least the finish time of activity l, i.e. $f_i \geq f_l$.
- Model 2 (cash outflows occur in a stepwise manner during the duration of activity i): the required delay should be in–between the delays for model 1 and 3: $f_i - X \geq f_l$, with $X \in \{1; d_i\}$. The value for X determines the part of activity i that has to be delayed after time t, in order to reduce the capital shortage at time t by the largest amount possible.

If no such activity l exists, the capital conflict at time t cannot be solved. In that case, $\Delta_{i,next} = +\infty$, the procedure terminates, penalty function (5.17) is applied and the solution is returned to the metaheuristic. If such an activity l does exist, the procedure compares $\Delta_{i,next}$ and $\Delta_{i,max}$ and either continues with *Delay until next positive* ($\Delta_{i,next} \leq \Delta_{i,max}$) or with *Delay until next successor* ($\Delta_{i,next} > \Delta_{i,max}$).

Delay until next positive: this part is applied if $\Delta_{i,next} \leq \Delta_{i,max}$, which implies that a sufficient delay to reach activity l found by the Set_i *calculations* part exists. The goal of *Delay until next positive* is to ensure that a delay of at least $\Delta_{i,next}$ is achieved. This way, the capital feasibility at time t can be improved since the negative cash flow of activity i is (partially) compensated by the positive cash flow of activity l. The set set_i is delayed by at least $\Delta_{i,next}$ time units, and subsequently scheduled at the earliest possible time for which the capital increases.

The method starts with a delay of $\Delta_{i,next}$ and checks the RR availability for all activities in set_i. In case a feasible delay is found, the procedure continues with the *Schedule* set_i part. Otherwise, $\Delta_{i,next}$ is incremented. The RR availability check continues until a feasible delay is found (continue with *Schedule* set_i) or until $\Delta_{i,next}$ becomes larger than $\Delta_{i,max}$. The latter case implies that no RR feasible delay of at least $\Delta_{i,next}$ could be found, in which case the procedure returns to the set *earlySet* of the *Check time t* part. The algorithm then continues with the next activity i in *earlySet* and repeats the Set_i *calculations* until a feasible delay is found or until *earlySet* is empty. If no next activity i in *earlySet* exists, the capital feasibility improvement procedure terminates, penalty function (5.17) is applied and the solution is returned to the metaheuristic.

Delay until next successor: this part is applied if $\Delta_{i,next} > \Delta_{i,max}$, which means that activity i cannot reach activity l obtained by the Set_i *calculations* part. The goal of *Delay until next successor* is to delay set_i as late as possible to reduce the capital shortage at time t. The set set_i is delayed by at most $\Delta_{i,max}$ time units, and scheduled at its latest possible time given this maximum delay.

The method starts with a delay of $\Delta_{i,max}$ and evaluates the RR availability for all activities in set_i. If a feasible delay is found, *Schedule* set_i is applied. If the delay is infeasible, $\Delta_{i,max}$ is decreased by 1 and the RR check is repeated. *Delay until next successor* terminates if a feasible delay is found (continue with *Schedule* set_i), or if $\Delta_{i,max}$ equals zero (return to the set *earlySet* of the *Check time* t part as done for the *Delay until next positive* part).

Schedule set_i: the fifth part is used if either *Delay until next positive* or *Delay until next successor* has found a feasible delay for set_i. The schedule is updated based on the delay, and single activities with a negative NPV are scheduled as late as possible. Once the *Schedule* set_i part has been completed, the procedure returns to the *Check time* t part, and searches again for periods t with a negative capital in order to construct a new set_i to shift.

5.4.2.3 NPV improvement

In this section, we briefly discuss the NPV improvement method. The activity move rules of this section correspond with the steps *Activity move rules 1* (network–based moves) and *Activity move rules 2* (schedule–based moves) in figure 5.8, and are adapted versions of the activity move rules of chapter 2 (Leyman and Vanhoucke, 2015) for the RCPSPDC.

Once a schedule is both *D–Feas* and *C–Feas* our algorithm aims to improve the project NPV while maintaining both deadline and capital feasibility. The goal of the NPV improvement move rules is to delay sets of activities with a negative cumulative NPV. Similar to the rules for capital feasibility improvement (section 5.4.2.2) we distinguish between a network– and schedule–based variant. The network–based delays consider an activity's predecessors and successors based on the project network, whereas the schedule–based delays consider the neighboring activities in the project schedule. We propose to first employ the network–based delays and then the schedule–based delays, in line with the results reported in chapter 2 for the RCPSPDC (Leyman and Vanhoucke, 2015). The only adaptation required for the CRCPSPDC, is a capital feasibility evaluation of every RR feasible delay. When the NPV improvement method is completed, the resulting schedule is returned to the metaheuristic.

5.5 Metaheuristics

In this section, we focus on the details of the metaheuristics used. We choose to compare the performance of three metaheuristics, namely a tabu search, a genetic algorithm and a scatter search. These three metaheuristics are selected since they allow for a comparison between single solution, single population and multi population approaches respectively. The specific implementation of each of the metaheuristics is based on three applications from literature, which report the best result to date for the problems studied in the respective papers. All three algorithms can be applied to both the CCPSPDC and the CRCPSPDC. The algorithm parameters are tested in section 5.6.2. We provide an overview of the application of each metaheuristic in the following paragraphs, but first discuss the solution representation used by all three algorithms.

Solution representation: since delays in sections 5.4.1 and 5.4.2 are done according to the order in which activities appear in the PL, we employ the topological order (TO) representation first proposed by Valls et al. (2004, 2003), and used for the RCPSP by Debels et al. (2006). The TO representation ensures a precedence feasible ordering of activities and is updated based on actual finish times after the improvement methods discussed in sections 5.4.2.2 and 5.4.2.3 have been applied. This way, a population element's initial PL is transformed into a finish time ordered activity list. Ties are broken randomly. For an example of the PL and TO representations, we refer to 5.A.1.

Tabu search (TS): we employ the TS of He et al. (2012) for the multi–mode capital–constrained project payment scheduling problem. The authors showed that the proposed TS performs best out of several alternatives. The initial solution's PL is generated randomly and is set as both the best and current solution. The appropriate scheduler of either section 5.4.1 or 5.4.2 is applied to evaluate the initial solution. Afterwards, a neighbor is generated from the current solution by applying a two–activity swap. If the neighbor's NPV (based on the scheduler used) is better than the best solution found, the latter is updated along with the current solution, and the reverse of the swap is added to the tabu list with length L. For the tabu list, the first–in–first–out principle is used, which means that once the list is full, the swap which has been in the list the longest is removed. If the neighbor's NPV is worse than the best solution found, the algorithm checks whether the swap employed is in the tabu list. If this is not the case, the current solution is updated and the reverse of the swap is added to the tabu list. Otherwise, the current solution is retained. Once the neighbor has been evaluated, and the current solution may have been updated, the TS again generates another neighbor of the current solution. If no change in the current solution has occurred after C neighbors have been generated, a new random PL is generated. New neighboring solutions are generated until a stopping criterion is

met.

Genetic algorithm (GA): we apply the GA proposed in chapter 2 for the RCPSPDC (Leyman and Vanhoucke, 2015). The GA obtained the best results to date and has subsequently been employed for the multi–mode RCPSPDC with different payment models in chapter 3 (Leyman and Vanhoucke, 2016b), based on extensive computational experiments. For the initial population $|P|$ PLs are randomly generated, with $|P|$ the GA's population size. The scheduler of either section 5.4.1 or 5.4.2 is employed for all elements. These $|P|$ elements are then all introduced in the GA's population P on which the selection, crossover and mutation operators are applied. A selection operator is used to select parents used for crossover. The elite selection operator is implemented, which randomly selects one parent from the subset R of the best solutions in the population, and uses four–tournament selection for the second parent. A one–point crossover is subsequently applied on the two selected parents to create two children. Afterwards, a two–activity swap mutation is imposed on each child with a probability of M, and the appropriate scheduler is employed. These steps for generating and updating children are repeated until $|P|$ children have been created. Once $|P|$ children have been generated, the best $|R|$ parents are retained and the rest of the parents are replaced by the best $|P| - |R|$ children. The set R is subsequently updated to contain the best elements in the population. Consider that this subset R is also the set from which the first parent is always selected in the elite selection operator. The selection, crossover and mutation operators are then again applied, and this is repeated until a stopping criterion has been met.

Scatter search (SS): the SS procedure used in this chapter is the algorithm of Van Peteghem and Vanhoucke (2011), which has the best results to date for the multi–mode RCPSP. First, an initial population with size $B \cdot (|B_1| + |B_2|)$ is randomly generated (*Diversification generation*). B_1 is the subset of best elements, whereas B_2 holds the most diverse elements found. B is a multiplication factor applied for the initial population. The scheduler of either section 5.4.1 or 5.4.2 is used to determine the NPV of each element from the initial population. Once the initial population has been generated, the subsets are constructed (*Subset generation*). Solutions are added to B_1 if their NPV is better than the best solution currently in B_1. Alternatively, if the solution NPV is better than the worst schedule in B_1, and if the minimum distance of the new solution to any solution in B_1 is greater than $v_1 \cdot n$ ($v_1 \in [0; 1]$), the new solution is also added to B_1. The minimum distance condition is imposed to ensure some degree of diversity in the set B_1. Solutions are added to B_2 if they are more diverse from any solution in B_1 than any other solution in B_2, or if the minimum distance of the new solution to any solution in B_1 is greater than $v_2 \cdot n$ ($v_2 \in [0; 1]$, $v_2 > v_1$). The distance between two solutions is calculated based on the following formula, similar to Van Peteghem and Vanhoucke (2011):

$$d^f_{p_1,p_2} = \sum_{i=1}^{n} |f_i^{p_2} - f_i^{p_1}| \tag{5.19}$$

p_1 and p_2 are the two solutions which are compared, and $f_i^{p_1}$ and $f_i^{p_2}$ their corresponding finish times for activity i. Additionally, if there are less elements in B_2 than $|B_2|$, the subset is seeded with random solutions. Solutions are subsequently combined by grouping all pairs from B_1 that contain at least one new solution, and all pairs with one solution from B_1 and one solution from B_2 (*Solution combination*). Two children are generated for every pair based on a one–point crossover. Each of the newly generated elements are scheduled (*Improvement method*), and afterwards both reference sets B_1 and B_2 are updated based on the same criteria discussed earlier (*Reference set update*). The generation of new elements and subsequent steps are repeated until a stopping criterion has been met.

5.6 Computational results

In this section, we discuss our computational results for both the CCPSPDC and the CRCPSPDC. We first give details of the proposed test data and subsequently configure the proposed schedulers and the three metaheuristics. We clearly show the added value of the steps of the capital feasibility improvement method. We analyze the influence of the data parameters and compare the results for the CCPSPDC with literature. Finally, we provide some managerial insights based on the trade–offs discussed in section 5.3.2. The stopping criterion based on 5,000 generated schedules as defined by Lova et al. (2009) is employed for all tests, and we assume a discount rate of 1%.

5.6.1 Test data

Project network: we use the same dataset as employed for the RCPSPDC by Vanhoucke (2010), which consists of 2,880 instances, but omit the renewable resources for the CCPSPDC. The parameter settings of the data are shown in table 5.3 and constitute the first five parameters, with deadline increase the percentage increase of the minimum project duration for the RCPSP. The project deadline is set by increasing the minimum project duration for the RCPSP with this percentage *D–Incr*.

Cash flows: we generate cash flow data for each of the 2,880 instances based on two parameters. Smith-Daniels and Smith-Daniels (1987) and Smith-Daniels et al. (1996) state that a project which does not generate capital will not be executed from a capital management point of view, which implies that the total cash inflows of a project have to exceed its total cash outflows. A profit margin percentage (*PMP*) parameter is used

Parameter	Values	Source
Number of activities (Act)	25, 50, 75 or 100	Vanhoucke (2010)
Order strength (OS)	0.25, 0.50 or 0.75	Vanhoucke (2010)
Resourse usage (RU)	2 or 4	Vanhoucke (2010)
Resource constrainedness (RC)	0.25, 0.50 or 0.75	Vanhoucke (2010)
Deadline increase (D–$Incr$)	5, 10, 15 or 20	Vanhoucke (2010)
Profit margin percentage (PMP)	0.33, 0.50 or 0.67	This chapter
Cash flow distribution (CFD)	0.33, 0.50 or 0.67	This chapter
Capital constrainedness (CC)	0.25, 0.50 or 0.75	This chapter

Table 5.3: Parameter settings of test instances.

to determine the difference between the total cash in– and outflows. The values for the PMP parameter are 0.33, 0.50 and 0.67, which correspond with a low, medium and high profit margin respectively. We start from the cash flow data of Vanhoucke (2010) for the RCPSPDC with 100% negative cash flows and apply the PMP on this data to generate a total cash inflow.

The distribution of the cash inflows over the activities in the project network also has an impact on the capital feasibility. As such, we introduce a cash flow distribution (CFD) parameter, which determines the distribution of the cash inflows over the activities in the project. The value of the CFD is set to 0.33, 0.50 or 0.67 which implies that respectively 33% (67%), 50% (50%) or 67% (33%) of the total cash inflow is evenly distributed over the first (second) $n/2$ activities. The division of activities in the first or second half of the project is based on the activity numbers. A low value for the CFD means that the cash inflows received by the first (second) $n/2$ activities are relatively small (large), whereas those received by the second $n/2$ activities are relatively large (small). The cash inflow for each of the first $n/2$ activities is equal to $-CFD \cdot \sum_{i=1}^{n} c_{i,out} \cdot \frac{1}{n/2}$, whereas the second $n/2$ activities' cash inflow equals $-(1 - CFD) \cdot \sum_{i=1}^{n} c_{i,out} \cdot \frac{1}{n/2}$. Based on the settings for both the PMP and CFD parameters, 9 combinations can be generated for each of the 2,880 instances.

Initial capital: we define a new parameter capital constrainedness (CC), inspired by the definition of resource constrainedness (RC) (Patterson, 1976), as follows:

$$CC = -\frac{\sum_{i=1}^{n} c_{i,out}/n}{C_0} \tag{5.20}$$

The enumerator of the right hand side holds the average of the cash outflows of all the activities in the project, whereas the denominator is the initial capital C_0. Based on a value set for the CC on the left hand side, the value for C_0 can be calculated. As an example, assume a project with 10 activities and a total cash outflow of -1,000. We wish to set the CC to 0.50. In this case, the initial capital C_0 is set to -(-1,000/10)/0.50 = 200.

Similar to the values chosen for the *RC* in the data of Vanhoucke (2010), we choose to employ values of 0.25, 0.50 or 0.75 for the *CC*. Finally, consider that the *CC* is independent of both the *PMP* and the *CFD*, since different values of both cash flow parameters change the cash inflows of the activities but not the cash outflows.

An overview of the values of the three additional data parameters *PMP*, *CFD* and *CC* can be found at the bottom of table 5.3. As a result, the total number of test instances is 2,880 x 9 cash flow variants x 3 *CC* levels = 77,760 instances per cash flow model. The 9 cash flow files for each project instance of Vanhoucke (2010) can be found online at *www.projectmanagement.ugent.be* (Research → Project scheduling → Net present value), along with the network data and the best known solutions for the CCPSPDC and the CRCPSPDC.

5.6.2 Algorithm configuration

In this section, we discuss the results of the configuration of our algorithm. We first briefly discuss the settings of the algorithm parameters, and subsequently go into detail about the added value of the proposed scheduler for both the CCPSPDC and the CRCP-SPDC. The performance of the three metaheuristics is also analyzed for both problems. All tests are run on 20% of the data presented in section 5.6.1, by employing each first out of five instances.

			PMP					
			0.33		0.50		0.67	
			Y_1	Y_3	Y_1	Y_3	Y_1	Y_3
Model 1	*CFD*	0.33	0.99	0.85	0.95	0.90	0.70	0.90
		0.50	0.95	0.90	0.75	0.95	0.70	0.95
		0.67	0.85	0.95	0.70	0.95	0.70	0.95
Model 2	*CFD*	0.33	0.99	0.85	0.95	0.80	0.70	0.80
		0.50	0.85	0.90	0.75	0.85	0.65	0.85
		0.67	0.80	0.95	0.65	0.90	0.60	0.90
Model 3	*CFD*	0.33	0.99	0.85	0.90	0.90	0.80	0.90
		0.50	0.90	0.90	0.80	0.95	0.75	0.95
		0.67	0.80	0.95	0.75	0.95	0.70	0.95

Table 5.4: Penalty function parameters.

5.6.2.1 Algorithm parameters

Y_2 of penalty function (5.18) is set to 20,000, a value much larger than the NPV of any feasible solution, whereas the values for Y_3 can be found in table 5.4, along with the best found values for the Y_1 parameter of penalty function (5.17) for the capital feasibility.

Based on our tests, we found that both values for Y_1 and Y_3 depend on *PMP* and on *CFD*, although the impact is larger for Y_1. Furthermore, minor differences can be observed in the values of both Y_1 and Y_3 between the three cash flow models. The values for the penalty function parameters are applicable to all three metaheuristics.

The best found values for the parameters of each of the metaheuristics are displayed in table 5.5. These values are, in general, in line with the values reported by the three applications of the algorithms in literature (He et al., 2012, Leyman and Vanhoucke, 2015, and Van Peteghem and Vanhoucke, 2011 respectively).

	Parameter	**Value**
TS	L	20
	C	10
GA	$\|P\|$	50
	$\|R\|$	5
	M	0.95
SS	B	9
	$\|B_1\|$	10
	$\|B_2\|$	8
	v_1	0.30
	v_2	0.50

Table 5.5: Metaheuristic parameters.

5.6.2.2 CCPSPDC

NPV improvement: we evaluate the added value of the NPV improvement, which is applied if the initial schedule is *C–Feas*. We compare the scheduler without the NPV improvement ($NPV\_$) and the scheduler with the NPV improvement of chapter 2 (Leyman and Vanhoucke, 2015) (NPV_L) in table 5.6. In both cases the GA is used as metaheuristic. The comparison between both alternatives is done based on the percentage average deviation from an upper bound (*%AvDev*). The lower the percentage deviation, the better the performance of the algorithm. The upper bound is calculated by the optimal procedure of Vanhoucke et al. (2001b) for the max–NPV problem. Only the instances for which both alternatives found a feasible solution are included. The percentage of *C–Feas* solutions found is displayed in the *%C* columns. We also compare the results based on the percentage average difference between the NPV in both cases (*%AvDiff*), and calculate the p–values (p) of the difference between the two options. Based on the results in the table it can be stated that the NPV improvement method has a clear added value for all three models, with the highest added value achieved for model 1.

Metaheuristics: in table 5.7 we compare the three metaheuristics. Only instances for

	NPV_-		NPV_L		ΔNPV	
	%AvDev	*%C*	*%AvDev*	*%C*	*%AvDiff*	p
Model 1	11.37	94.10	**5.35**	94.13	7.38	<0.001
Model 2	7.65	96.14	**2.25**	96.12	6.35	<0.001
Model 3	4.37	98.57	**0.07**	98.59	4.85	<0.001

Table 5.6: Added value NPV improvement CCPSPDC.

which all alternatives found a feasible solution are taken into account. The GA is shown to perform best ($p < 0.001$) for all three models, although it can be observed that the difference between the three alternatives is very small for model 3.

	TS		**GA**		**SS**	
	%AvDev	*%C*	*%AvDev*	*%C*	*%AvDev*	*%C*
Model 1	5.31	92.91	**5.00**	**94.13**	5.30	93.08
Model 2	2.19	94.73	**1.98**	**96.12**	2.20	95.27
Model 3	0.07	98.09	**0.06**	**98.75**	0.07	98.25

Table 5.7: Comparison of metaheuristics CCPSPDC.

Computation times: table 5.8 provides an overview of the average computation times in seconds (s) for the three models, given the 5,000 schedules stopping criterion. We distinguish between the average computation time used by the scheduler ($AvCT_S$), the time used by the metaheuristic ($AvCT_M$), and the total computation time ($AvCT_T$). The numbers between brackets display the percentage of the total computation used for the scheduler, metaheuristic and the total procedure respectively. The results in the table indicate that the majority of the computation time is spent applying the scheduler to solutions generated by the GA. The average computation times are the highest for model 2 due to the more complex cash flow profiles (section 5.3.1).

	CCPSPDC			**CRCPSPDC**		
	$AvCT_S$	$AvCT_M$	$AvCT_T$	$AvCT_S$	$AvCT_M$	$AvCT_T$
Model 1	1.04	0.16	1.21	4.78	0.16	4.94
	(86.37)	(13.63)	(100.00)	(96.82)	(3.18)	(100.00)
Model 2	1.41	0.16	1.58	7.87	0.17	8.04
	(89.57)	(10.43)	(100.00)	(97.87)	(2.13)	(100.00)
Model 3	0.66	0.16	0.82	4.45	0.19	4.64
	(80.37)	(19.63)	(100.00)	(95.83)	(4.17)	(100.00)

Table 5.8: Computation times (s).

5.6.2.3 CRCPSPDC

Capital feasibility improvement: we test the added value of each part of the capital feasibility improvement, by comparing several options based on the percentage of *C–Feas* solutions found (*%C–Feas*) in table 5.9, with the best option marked in bold. The alternatives which we consider are the following: C_{--}: no capital feasibility improvement; C_{SD}: only the schedule–based moves (S) and the delay of activities with a negative NPV (D); C_{ND}: only the network–based moves (N) on top of D; C_{FD}: the full proposed method (F) with first the schedule– and then the network–based moves, including the delay of activities with a negative NPV; C_{RD}: first the network– and then the schedule–based moves (reverse: R); C_{F-}: the proposed method without the delay of activities with a negative NPV.

	C_{--}	C_{SD}	C_{ND}	C_{FD}	C_{RD}	C_{F-}
Model 1	43.94	78.86	77.00	**80.19**	79.31	76.36
Model 2	57.58	86.94	85.80	**87.77**	87.10	86.16
Model 3	95.22	97.39	97.36	**97.42**	97.36	97.30

Table 5.9: Added value capital feasibility improvement CRCPSPDC (*%C–Feas*).

Based on the results in table 5.9, we conclude that the proposed capital feasibility improvement method, with first the schedule– and then the network–based moves, performs best for all three models. As illustrated in 5.A.1, the schedule–based variant of the improvement method offers a more efficient way to reduce capital shortages, which leads to a higher capital feasibility. Hence, the schedule–based delays should be applied first for all three models. The added value of delaying single activities with a negative NPV is made clear as well by comparing the *%C–Feas* values for the C_{FD} and C_{F-} options. Based on the higher feasibility for C_{FD}, we conclude that the single activity delays have an added value by allowing more capital shortages to be solved (5.A.2). It can, however, be observed that for model 3 only small differences can be found between the different options for the capital feasibility improvement method, but that the method allows for an improvement nonetheless when compared to C_{--}.

NPV improvement: the proposed scheduler without any NPV improvement (NPV_-) is compared with the scheduler with the NPV improvement of chapter 2 (Leyman and Vanhoucke, 2015) (NPV_L) in table 5.10, based on the GA. Only instances for which a feasible solution could be found by both NPV_- and NPV_L, are included in the *%AvDev* and *%AvDiff* calculations in the table. *%D* is the percentage of deadline feasible solutions found. Based on the results in the table, we conclude that the NPV improvement method has a significant, albeit small, added value for all three models. The added value is rela-

tively small due to the rules for capital feasibility improvement delaying sets of activities, independent of the set's cumulative NPV. As such, sets of activities with a negative cumulative NPV may already have been delayed, in particular by the *Delay until next successor* part of the capital feasibility improvement method of section 5.4.2.2 and by the *Delay negatives* step.

	NPV_-			NPV_L			ΔNPV	
	%AvDev	*%C*	*%D*	*%AvDev*	*%C*	*%D*	*%AvDiff*	*p*
Model 1	33.59	80.19	98.86	**33.56**	80.39	99.01	0.08	<0.001
Model 2	31.88	87.77	99.02	**31.81**	87.13	99.11	0.18	<0.001
Model 3	30.34	97.45	99.02	**30.30**	97.58	99.09	0.05	<0.001

Table 5.10: Added value NPV improvement CRCPSPDC.

Metaheuristics: in table 5.11 we compare the performance of the three metaheuristics. We also compare with 10,000 random schedules. Only instances for which all alternatives found a feasible solution are taken into account. We conclude that the GA performs best for all three models (lowest *%AvDev* and highest *%C*), with the SS second best. Additionally, all p–values, based on a pairwise comparison between each of the four alternatives, are smaller than 0.001, indicating a significant difference. Only the difference between TS and 10,000 randoms had a slightly bigger p–value of 0.004 for model 1.

	TS		**GA**		**SS**		**10,000 randoms**	
	%AvDev	*%C*	*%AvDev*	*%C*	*%AvDev*	*%C*	*%AvDev*	*%C*
Model 1	38.02	72.99	**33.62**	**80.39**	35.98	74.49	38.02	72.49
Model 2	35.36	79.69	**31.49**	**87.13**	33.60	83.11	35.39	80.74
Model 3	33.20	92.26	**29.91**	**97.58**	31.69	95.36	33.29	91.79

Table 5.11: Comparison of metaheuristics CRCPSPDC.

The differences in performance between the three metaheuristics (tables 5.7 & 5.11) can be explained along the following lines:

- The TS is a single solution–based metaheuristic, whereas the GA and SS are population–based metaheuristics. As a result, there is no interaction between different solutions in the TS, whereas the GA and SS on the contrary allow for interaction by making use of a crossover operator, resulting in a better performance for the latter.
- A TS algorithm may be too local and can have issues with respect to diversity as a result (Gendreau and Potvin, 2010).
- Based on section 5.5, it can be stated that the GA employs an elite set and as a result requires a higher mutation rate (Reeves, 2010). This way, the GA implicitly

distinguishes between a high quality set and between a diverse set, similarly to the SS algorithm. The main difference with the SS is the lack of distance functions to permit entrance into both sets.

- The use of the elite set in the GA bears resemblances to evolutionary path relinking, namely the combination of high quality solutions (Resende et al., 2010).

- Based on these final two observations, we can state that the GA contains elements of both a SS and of a path relinking algorithm, but that the inclusion of these elements in the GA framework leads to better results than in the SS framework.

To mitigate these issues, and to show the validity of our explanation, we have adjusted the three metaheuristics in the following way:

- In the TS we implement some form of interaction by applying a one–point crossover. The generated child is used as new starting solution, whereas the father is the best results obtained after the previous run of 500 generations, and the mother is a randomly generated element. Furthermore, for generating a neighboring solution, the TS now applies a two–activity swap operator repeatedly, in order to increase diversity. Our tests indicate that a factor 4 is appropriate. The resulting algorithm is referred to as TS' in the remainder of this section.

- The GA' algorithm is more similar to the "standard" GA framework (Reeves, 2010), and uses a four–tournament selection for both parents instead of the elite selection. A lower mutation rate of 5% is also used, along with a population size of 90 and only the best element is retained after each generation. These parameter settings have been tested and displayed the best results.

- The SS' algorithm omits the distance function on the diverse set B_2 and applies a two–activity swap with a mutation rate of 10% on all new elements generated with one parent from the high quality set B_1 and one parent from B_2. This way, the diversity is amplified in the SS' algorithm in comparison to the SS algorithm. In doing so, we introduce several elements more typical for a GA in the SS framework.

All additional tests with respect to the TS', GA' and SS' algorithms have been run for the CRCPSPDC with model 2, since this model can be seen as a middle ground in the general model of section 5.3. A summary of the results is displayed in table 5.12, which shows the differences between the six metaheuristics based on $\%AvDev$, the average project NPV of the feasible instances found by all alternatives ($AvNPV$), and $\%C$. We can conclude that:

- The TS' and SS' have a better performance than the TS and SS respectively, which validates our claims with respect to the shortcomings in the TS and SS frameworks for the problem under consideration.
- The results of the GA' are worse than those of the GA, which highlights the added value of including some SS and path relinking concepts in the GA.
- GA' still obtains better results than the improved SS', which shows that the GA framework allows for a higher performance than the SS framework on the CRCP-SPDC. In chapter 2 (Leyman and Vanhoucke, 2015), similar results were obtained when comparing the GA and a more "standard" GA implementation with the SS of Vanhoucke (2010) for the RCPSPDC. It can furthermore be stated that a GA typically employs randomization, whereas a SS uses memory–based strategies (Martí et al., 2006), which shows that randomization is more important given the NPV objective and the proposed schedulers.

	TS	**TS'**	**GA**	**GA'**	**SS**	**SS'**
%AvDev	35.40	35.24	31.53	32.00	33.64	32.99
AvNPV	3,831.74	3,853.80	4,109.67	4,074.26	3,954.48	3,996.73
%C	79.69	80.59	87.13	87.31	83.11	84.37

Table 5.12: Additional comparison of metaheuristics CRCPSPDC (model 2).

Computation times: an overview of the average computation times for the CR-PCPSDC is included in the right part of table 5.8. The conclusions are similar as for the CCPSPDC (left part of the table), and show that the majority of the computation time is used for the scheduler, and that the procedure is the slowest for model 2. The computation times for the scheduler are considerably higher for the CRCPSPDC compared to those for the CCPSPDC, which was to be expected given the more complex scheduler due to the additional resource limitations.

Convergence: we analyze the convergence of the GA based on the number of generations until the 5,000 schedules stopping criterion is reached. Figure 5.9 displays the average project NPV for the three models as the number of generations of the GA increases for the CRCPSPDC. The average number of generations is 54, and an average of 1.85 schedules are used for a single application of the scheduler.

5.6.3 Discussion & comparison

In this section, we discuss the best results for the CCPSPDC (table 5.13) and for the CRCPSPDC (table 5.14). We compare the results for the CCPSPDC with a composite

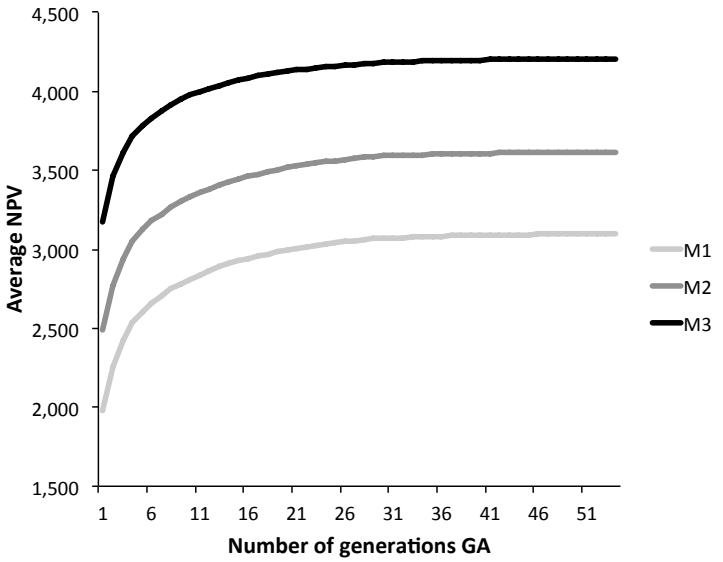


Figure 5.9: Convergence GA.

heuristic, which is based on two algorithms from literature. The composite algorithm splits the problem into two parts, namely optimize the project NPV and reduce capital shortages, similar to Smith-Daniels et al. (1996). The proposed heuristic employs the best method of Smith-Daniels et al. (1996) to make decisions with respect to capital, but uses the exact recursive method of Vanhoucke et al. (2001b) for the max–NPV problem. In the remainder of this section, the composite procedure is abbreviated as SD–V, or Smith–Daniels–Vanhoucke.

From the results in tables 5.13 and 5.14 the following general conclusions can be drawn:

- The GA outperforms the SD–V algorithm for all three models (table 5.13). The p–values of the comparison between both algorithms are smaller than 0.001 for the three models.

- Capital feasibility is the hardest to obtain with a small or a large number of activities. For a low number of activities, i.e. 25, only a small number of possible sequences exist in which the activities can be scheduled. This implies that capital shortages are harder to solve, because fewer activities can be moved to reduce shortages. For a high number of activities, i.e. 100, a great many ways exist in which activities can be delayed. Hence, it is more difficult to find the best delays to solve capital shortages.

- The impact of the distribution of the cash inflows over activities (CFD) has a larger impact on the project NPV and on capital feasibility, compared to the profit of the

project (PMP).

- The inclusion of renewable resources makes the problem harder to solve (higher $\%AvDev$ and lower $\%C$ when comparing tables 5.13 and 5.14). In the CRCPSPDC (table 5.14), imposing a stronger RR restriction (a higher RU and RC value) makes it easier to construct a *C–Feas* schedule, but at the cost of a lower project NPV.

- The computation times in table 5.14 give an indication of the drivers of algorithm complexity in terms of the data parameters. The number of activities and the resource usage have the largest impact.

		Model 1				**Model 2**				**Model 3**			
		GA		**SD–V**		**GA**		**SD–V**		**GA**		**SD–V**	
		%AvDev	*%C*	*%AvDev*	*%C*	*%AvDev*	*%C*	*%AvDev*	*%C*	*%AvDev*	*%C*	*%AvDev*	*%C*
Act	25	3.50	87.84	5.51	79.62	1.41	92.25	2.39	88.19	0.06	98.33	0.10	97.67
	50	3.84	97.13	6.33	93.13	1.43	98.80	2.49	96.98	0.03	99.99	0.01	99.94
	75	4.86	97.85	7.86	94.28	1.85	99.09	3.15	96.93	0.05	99.96	0.06	99.67
	100	6.01	95.07	9.65	89.76	2.58	96.08	4.33	92.16	0.08	96.27	0.14	95.45
OS	0.25	4.49	98.61	6.75	95.94	1.91	99.78	2.88	98.96	0.04	100.00	0.01	99.98
	0.50	4.95	95.70	8.13	88.92	1.92	98.06	3.38	93.74	0.06	99.54	0.09	99.26
	0.75	4.29	89.10	7.32	82.75	1.60	91.82	3.01	87.99	0.07	96.37	0.15	95.30
D–Incr	5	4.54	92.87	7.33	86.41	1.79	95.95	3.04	92.15	0.05	98.55	0.08	98.14
	10	4.56	94.24	7.35	88.80	1.81	96.51	3.08	93.46	0.05	98.65	0.08	98.19
	15	4.59	95.12	7.41	90.25	1.83	96.79	3.11	94.19	0.06	98.68	0.08	98.19
	20	4.62	95.66	7.46	91.35	1.84	96.97	3.13	94.47	0.06	98.66	0.08	98.21
PMP	0.33	4.98	90.22	8.42	82.48	2.20	93.12	3.91	87.58	0.12	96.12	0.18	95.14
	0.50	4.80	95.57	7.64	90.61	1.88	97.52	3.15	95.05	0.04	99.79	0.06	99.43
	0.67	4.02	97.64	6.24	94.51	1.42	99.02	2.30	98.06	0.01	100.00	0.00	99.98
CFD	0.33	8.89	85.44	14.98	74.42	3.73	90.05	6.74	82.33	0.10	95.91	0.24	94.55
	0.50	3.32	98.48	5.24	95.75	1.14	99.72	1.90	98.95	0.02	100.00	0.00	100.00
	0.67	2.53	99.50	3.71	97.43	0.91	99.90	1.25	99.41	0.05	100.00	0.00	100.00
CC	0.25	2.51	98.78	4.60	96.34	0.64	99.41	1.33	98.47	0.04	99.49	0.02	99.21
	0.50	5.06	94.18	8.04	88.14	1.96	96.50	3.41	93.01	0.05	98.58	0.09	98.27
	0.75	6.47	90.46	9.92	83.11	2.97	93.75	4.70	89.22	0.07	97.84	0.13	97.06
Overall		4.58	94.47	7.39	89.20	1.82	96.55	3.09	93.56	0.06	98.64	0.08	98.18

Table 5.13: Comparison with literature CCPSPDC.

		Model 1			Model 2			Model 3		
		%AvDev	*%C*	*AvCT (s)*	*%AvDev*	*%C*	*AvCT (s)*	*%AvDev*	*%C*	*AvCT (s)*
Act	25	16.89	73.21	0.55	15.37	83.07	0.71	14.45	97.67	0.49
	50	28.85	83.80	2.67	27.13	91.52	4.00	25.93	99.52	2.20
	75	38.58	83.20	6.04	36.59	91.18	9.11	35.71	98.72	5.02
	100	47.45	78.81	12.34	45.59	85.96	18.92	44.61	94.35	10.15
OS	0.25	32.71	85.64	6.78	31.17	93.60	10.88	30.53	98.84	5.52
	0.50	35.06	79.00	5.25	32.89	87.84	7.87	31.32	98.16	4.34
	0.75	31.92	74.62	4.17	29.80	82.36	5.80	28.23	95.70	3.53
RU	2	26.79	72.90	2.02	24.47	84.80	2.97	23.19	97.53	1.87
	4	38.66	86.61	8.78	37.69	91.06	13.40	36.90	97.60	7.05
RC	0.25	32.16	76.37	4.79	29.84	85.95	7.17	28.22	97.24	4.02
	0.50	33.61	80.64	5.52	31.84	88.47	8.37	30.71	97.67	4.50
	0.75	33.87	82.26	5.90	32.22	89.37	9.01	31.19	97.79	4.88
D–Incr	5	36.25	65.41	3.97	33.03	77.54	5.98	30.79	94.24	3.44
	10	33.51	80.24	5.06	31.44	89.38	7.59	30.09	98.55	4.30
	15	32.32	85.31	5.93	30.71	91.78	8.79	29.75	98.71	4.72
	20	31.64	88.05	6.65	30.37	93.02	10.37	29.60	98.76	5.39
PMP	0.33	32.47	74.59	6.63	30.57	82.50	10.54	29.64	94.86	5.58
	0.50	33.59	80.95	5.41	31.65	89.15	8.30	30.19	98.72	4.35
	0.67	33.58	83.73	4.16	31.67	92.15	5.71	30.29	99.12	3.46
CFD	0.33	49.21	64.10	4.82	45.59	73.26	7.63	40.33	94.42	4.01
	0.50	31.19	85.86	4.71	29.34	94.34	6.44	28.47	99.12	3.84
	0.67	23.74	89.29	6.67	22.39	96.19	10.48	21.83	99.16	5.54
CC	0.25	30.88	93.54	5.54	30.15	96.87	8.26	30.13	98.54	4.58
	0.50	33.82	77.82	5.50	31.35	87.51	8.46	29.98	97.54	4.54
	0.75	35.82	67.90	5.16	32.70	79.41	7.83	30.03	96.62	4.27
Overall		33.24	79.75	5.40	31.32	87.93	8.18	30.05	97.57	4.46

Table 5.14: Best results CRCPSPDC.

5.6.4 Managerial insights

In order to evaluate the different trade–offs for the CRCPSPDC (section 5.3.2), we generate additional project data with the network generator RanGen (Demeulemeester et al., 2003). The focus lies on testing the impact of the *OS*, *RC* and *CC* values on the project NPV and capital feasibility in more detail. An overview of the data parameters is given in table 5.15, with a total of 32,805 instances (= 5 instances x 9 *OS* levels x 9 *RC* levels x 9 *CC* levels x 3 *PMP* levels x 3 *CFD* levels), applicable to each of the three models. The cash flow data, with three levels for the *PMP* and *CFD* factors, is generated similarly as done in section 5.6.2. The additional instances generated are available online at *www.projectmanagement.ugent.be*, along with the cash flow data and reported results for each instance.

Parameter	**Values**
Number of activities (Act)	30
Resourse usage (RU)	4
Deadline increase (D–$Incr$)	10
Profit margin percentage (PMP)	0.33, 0.50 or 0.67
Cash flow distribution (CFD)	0.33, 0.50 or 0.67
Order strength (OS)	0.10 to 0.90 in steps of 0.10
Resource constrainedness (RC)	0.10 to 0.90 in steps of 0.10
Capital constrainedness (CC)	0.10 to 0.90 in steps of 0.10

Table 5.15: Parameter settings insights.

We have applied an ANOVA to determine the effect of *OS*, *RC* and *CC* on the average project NPV ($AvNPV$) for the three models. All single–factor and two–factor cross effects have a p–value lower than 0.001, except for the combination of *RC* and *CC* for all models (p–values of nearly 1) and the cross effect of *OS* and *CC* for model 3 (p–value of 0.264). The three–factor term (OS*RC*CC) obtained a p–value of 1 for the three models and is not considered. Based on significant p–values for the single– and most of the two–factor effects, it is worthwhile to investigate the impact of the factors *OS*, *RC* and *CC* further. We continue the remainder of the analysis with model 2, since this model is a middle ground in the general model of section 5.3. Furthermore, aside from the aforementioned difference for model 3, the same conclusions apply to all three models, and by extension to the general model. We analyze the impact of the network structure (OS), the resource availability (RC), and the capital availability (CC) on the project NPV ($AvNPV$) and on the capital feasibility ($\%C$). The latter can be seen as a measure for the required capital management by the contractor, since a higher (lower) feasibility implies that it is easier (harder) to schedule the project given the available capital. Typically, a higher

NPV and higher capital feasibility are preferable for the contractor, because the project then has a greater contribution to the company profits and requires a smaller focus on the optimization of the available capital.

Figure 5.10 displays the single–factor effects of OS, RC and CC on $\%C$ and $AvNPV$, whereas figure 5.11 shows the two–factor cross effects. The following guidelines can be stated for the contractor:

- **Network**
  - The contractor can on average increase the project NPV and reduce the need for capital management by making a project more parallel (lower OS value). This way, cash inflows can be received earlier, resulting in a higher NPV and a lower initial capital requirement (figure 5.10).
  - In more parallel projects the effect of the renewable resource availability plays a greater role. More capital management is on average required in case the project has a high resource availability (lower RC value), whereas the contractor should pay more attention to the project NPV if fewer resource units can be used (top two graphs of figure 5.11).
  - In more serial projects with a relatively high initial capital (low CC value) the contractor should focus on properly optimizing the project NPV, whereas decreases in capital availability warrant on average a greater focus on capital management (middle two graphs of figure 5.11).
- **Resource availability**
  - A high resource availability reflects favorably on the project NPV but requires more capital management, whereas the opposite holds true for a low resource availability (figure 5.10).
  - Capital management is particularly crucial in case of a low initial capital and a large availability of renewable resources (bottom two graphs of figure 5.11).
- **Capital availability**
  - Lowering the initial capital mainly results in a higher degree of capital management, but has on average little effect on the project NPV (figure 5.10).
  - Increasing the initial capital decreases the project NPV on average, since less profitable projects can then be executed. The major question for the contractor then becomes whether the resulting project NPV is worth the required capital investment (figure 5.10).

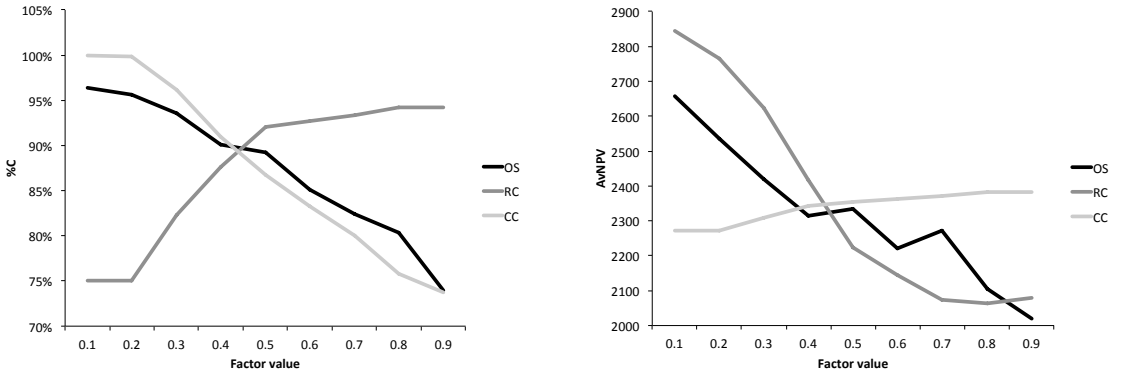


Figure 5.10: Impact of single–factor effects *OS*, *RC* and *CC* on *%C* and *AvNPV* (model 2).

5.7 Conclusions & future research

In this chapter, we have presented new scheduling techniques for the capital–constrained project scheduling problem with discounted cash flows (CCPSPDC) and for the capital– and resource–constrained project scheduling problem with discounted cash flows (CRCP-SPDC). The objective is to schedule the project activities to maximize the project net present value (NPV). Capital constraints impose that the sum of the cash inflows received and cash outflows paid, given an initial capital, cannot be negative at any point in time during the project duration. We apply three cash flow models, as part of a general model for the distribution of cash outflows, to both problems.

In order to solve capital shortages, activities can be delayed together with the set of succeeding activities. The distinction is made between two types of delays to reduce the capital shortage. The first type delays sets of activities such that cash outflows leading to a capital shortage are compensated by cash inflows of other activities. The second type delays activities until a succeeding activity is reached, in which case later iterations of the procedure allow for additional delays. Once the capital shortages have been solved, activity move rules are applied, which improve the project NPV by delaying sets of activities with a negative cumulative NPV. The proposed scheduling techniques have been implemented as part of three metaheuristics from literature, along with two penalty functions, one to improve deadline feasibility and another to improve capital feasibility.

Extensive computational experiments have shown the added value of the delay rules for both the capital feasibility improvement and for the NPV improvement. The best results for the CCPSPDC have been favorably compared with a composite method based on literature.

As a first future research avenue, the proposed models and scheduling techniques can be applied in a multi–mode context. In doing so, the contractor can decide between

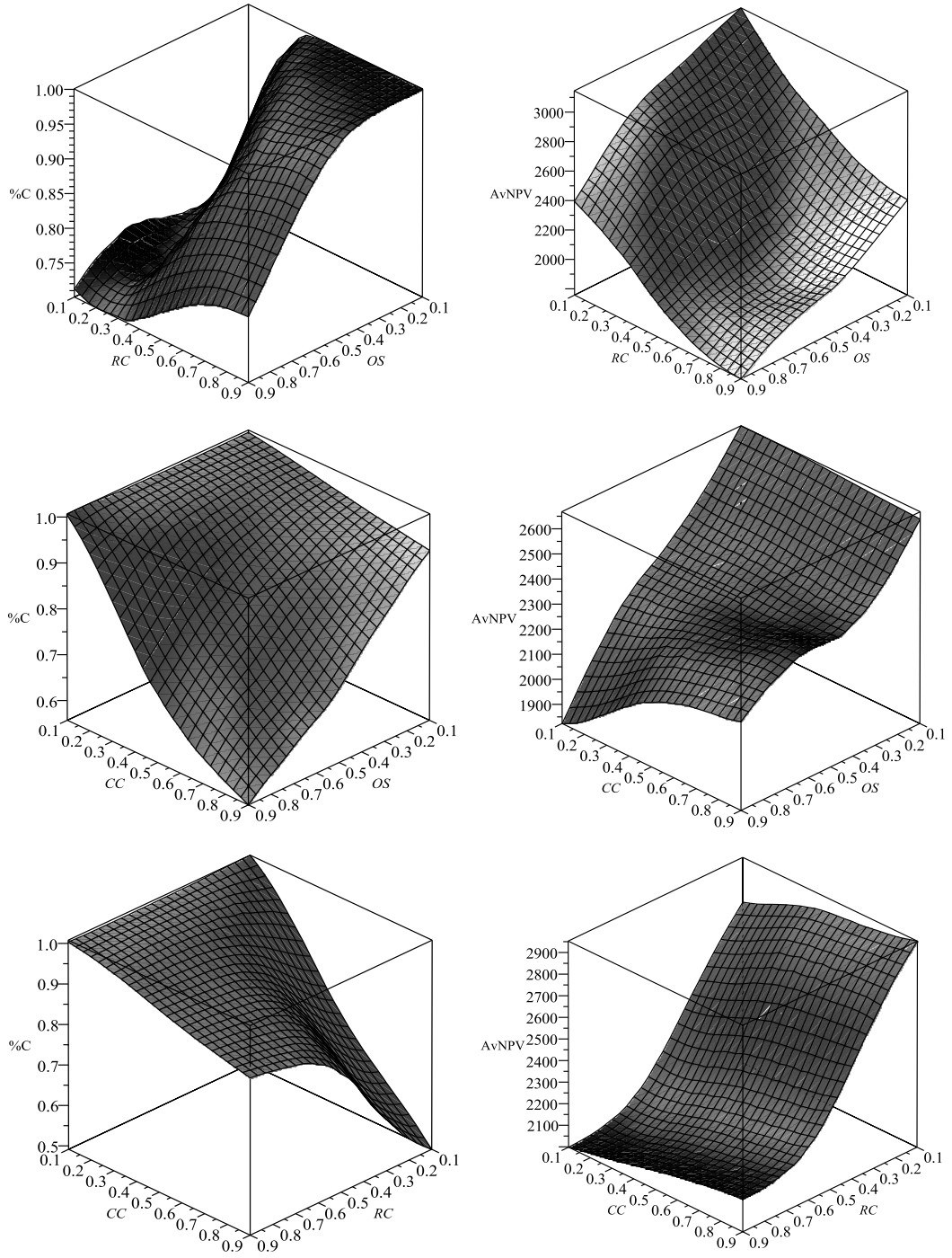


Figure 5.11: Impact of two–factor cross effects *OS*, *RC* and *CC* on *%C* and *AvNPV* (model 2).

different modes for each activity, and this allows for the inclusion of different applications per activity of the general model for cash outflows.

Second, whereas the focus of this manuscript has been on the cash outflows and capital restrictions, it may be interesting to extend these models to include different scenarios for the cash inflows as well (table 5.1).

Finally, including a client–contractor negotiation process would allow for the determination of the timing and size of the cash inflows based on interactions between both parties. Furthermore, the timing and size of cash outflows can be linked to the negotiation with subcontractors. This way, the entire interaction process between the three parties (i.e. client, contractor and subcontractor) can be integrated and optimized.

5.A Appendix

In this appendix, we illustrate the capital feasibility improvement method of section 5.4.2.2 with two examples. The first example shows the network– and schedule–based variants of the improvement method, whereas the second example details the added value of delaying activities with a negative NPV for the capital feasibility.

5.A.1 Example 1

Example data: the example of figure 5.12 is used to illustrate the concepts of the capital feasibility improvement method for model 3. We make use of a single RR with availability of 5, assume a deadline of 19, and an initial capital C_0 of 20. The discount rate is set to 1%. The initial schedule of the example is constructed based on the PL (1, 3, 5, 2, 4, 6, 7, 10, 9, 8, 11, 12, 13). Activity 4 is delayed by 1 time unit due to its negative net cash flow. The resulting initial schedule can be found on the right in figure 5.12, with the time on the horizontal axis, the RR level a_1 on the left vertical axis and the capital C_t at time t on the right vertical axis. The bold line indicates the available capital during the project duration, with decreases and increases corresponding with cash out– and inflows respectively. The capital at time 0 equals the initial capital of 20, whereas the capital at each time t is equal to the sum of C_0 and the net cash flows of all activities completed no later than time t. E.g. C_5 = 20 (C_0) - 15 ($c_{2,net}$) - 25 ($c_{4,net}$) + 5 ($c_{5,net}$) = -15. The capital at time 19 is the sum of C_0 and the cash flows of all activities and equals 60. Based on this schedule it is clear that a capital shortage exists of 15 at times 5 and 6, which leads to an *ECR* value equal to 30.

Network–based moves: the network–based capital feasibility improvement is first applied. The *Check time t* part leads to an *earlySet* consisting of activities 4 and 2 at time 5. Activity 4 occurs last in the PL, so this activity is used first for the Set_i *calculations* part. The corresponding set set_4 consists of activities 4 and 7. The earliest cash inflow

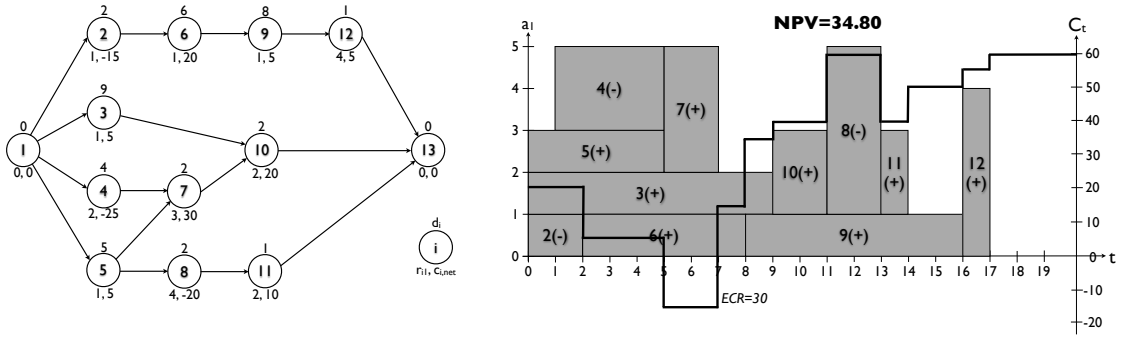


Figure 5.12: Data and initial schedule example 1.

to occur after $f_4 = 5$ is that of activity 6 at time 8. As such, $\Delta_{4,next}$ equals 3 (= 8 - 5). $\Delta_{4,max}$ is determined by the successors not in set_4 of both activities 4 and 7. Only activity 10 meets these criteria and with its finish time of 11 this leads to $\Delta_{4,max}$ being equal to 2. Since $\Delta_{4,max}$ is smaller than $\Delta_{4,next}$, the *Delay until next successor* part of the capital feasibility improvement method is used next. The RR availability is evaluated for a delay of $\Delta_{4,max}$ by set_4, which is feasible. Hence, both activities 4 and 7 are delayed by the *Schedule* set_i part with $\Delta_{4,max}$ time units, and the capital and RR levels are updated. Finally, the *Schedule* set_i part checks whether any activities with a negative NPV can be delayed, but this is not the case.

The procedure returns to the *Check time t* part and once again evaluates the capital feasibility at time 5, based on the left schedule of figure 5.13. Since the capital level at time 5 is positive t is incremented to 6, at which the available capital is also non–negative. The procedure continues to time 7, which has a negative capital so *earlySet* is constructed and consists of activities 4 and 2. Set_i *calculations* is applied for activity 4 and set_4 consists of activities 4, 7 and 10. $\Delta_{4,next}$ is calculated based on activity 6 and equals 1. $\Delta_{4,max}$ equals 8 since the only successor not in set_4 of any of the three activities is the end dummy. *Delay until next positive* is applied but the method cannot find any RR feasible delay for set_4. As such the capital feasibility improvement method backtracks to *earlySet* in the *Check time t* part. Set_i *calculations* is done for activity 2 which results in set_2 consisting of activities 2, 6, 9 and 12. $\Delta_{2,next}$ is determined based on activity 6 and equals 6, whereas $\Delta_{2,max}$ is 2 due to the project deadline. The *Delay until next successor* part is applied, but no feasible delays can be found. The procedure returns to *earlySet*, but all activities in the set have been considered, so the capital feasibility improvement method terminates since the capital shortage at time 7 could not be solved. The resulting schedule after the network–based moves of the capital feasibility improvement method have been applied is shown in the left schedule of figure 5.13. The capital shortage has both been reduced (*ECR*

equal to 15 instead of 30) and moved later in time, but the project NPV has decreased due to the delay of a set of activities with a positive cumulative NPV (activities 4 and 7).

Schedule–based moves: the schedule–based variant of the capital feasibility improvement method is started based on the left schedule of figure 5.13. For the capital shortage at time 7, *earlySet* consists of activities 4 and 2. Set_i *calculations* is first applied for activity 4 since this activity appears after activity 2 in the PL. The set set_4 holds activities 4, 7, 10, 8 and 11, the last two of which would never have been included in the network–based version of the improvement method. $\Delta_{4,next}$ equals 1 based on the positive cash flow of activity 6 and $\Delta_{4,max}$ is 5 because of the end dummy activity 13 as successor of activity 11. Since $\Delta_{4,next}$ is smaller than $\Delta_{4,max}$, the procedure continues with the *Delay until next positive* part. A delay of 1 time unit is RR feasible for set_4, so the *Schedule* set_i part is used next. The activity finish times for all activities in set_4 are updated, as are the capital and RR levels. No activities with a negative NPV can be delayed as part of *Schedule* set_i. The procedure returns to part *Check time t*, but no capital shortages remain so a *C–Feas* schedule has been found and the method terminates. The schedule after both variants of the capital feasibility improvement method have been applied can be found in the right schedule of figure 5.13.

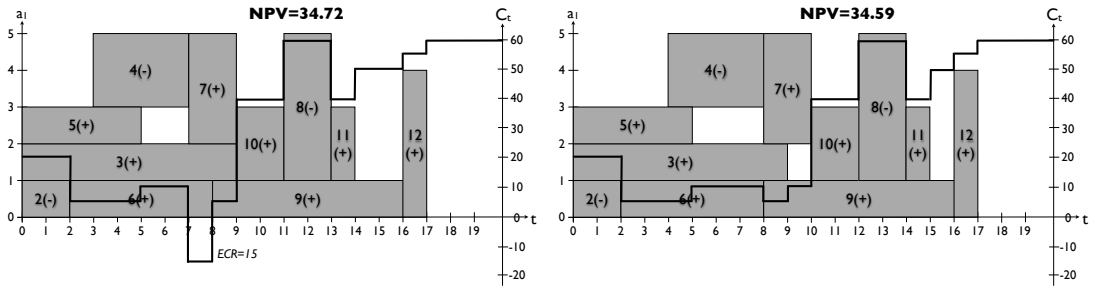


Figure 5.13: Schedules example 1 after network– & schedule–based capital feasibility improvement.

Solution representation: recall that the PL of the schedule in figure 5.12 was (1, 3, 5, 2, 4, 6, 7, 10, 9, 8, 11, 12, 13). Based on the right schedule of figure 5.13, the updated list, or TO, becomes (1, 2, 5, 6, 4, 3, 7, 10, 8, 11, 9, 12, 13), with the tie between activities 4 and 6 broken randomly in favor of activity 6.

Alternative: assume that the schedule–based variant of the capital feasibility improvement is first applied for the same example. We again start from the schedule in figure 5.12. Activities 4 and 7 are first delayed by 2 time units in the same manner as done before. The available capital at time 7 is, however, negative and *earlySet* consists of activities 4 and 2. Unlike for the network–based improvement, a delay of activity 4, along with activities 7, 8, 10 and 11 is now possible. This delay of 1 time unit would not

have been considered in the network–based step. The resulting schedule is *C–Feas* and is displayed in figure 5.14. It is important to consider that the schedule–based moves are more efficient for the example, since fewer moves are required to obtain a *C–Feas* schedule. Our results of section 5.6.2 confirm that it is indeed best to first apply the schedule–based variant of the capital feasibility improvement method.

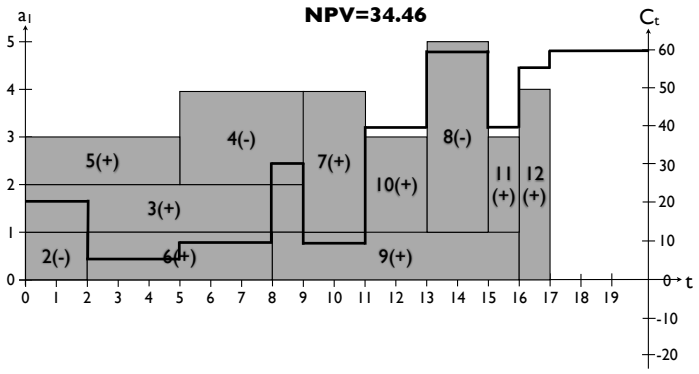


Figure 5.14: Alternative schedule example 1.

5.A.2 Example 2

In step 4 of section 5.4.2.1 and the *Schedule* set_i part of section 5.4.2.2, we stressed the need for delaying single activities with a negative NPV as much as possible given both the precedence relations and the RR constraints. Let us illustrate the reasoning behind these delays with the simple example for model 3 shown in figure 5.15. The project has a RR availability of 4, a deadline of 14 and an initial capital C_0 of 10. The PL used is (1, 3, 2, 4, 5, 6), and the discount rate is 1%. The top right schedule in figure 5.15 shows the initial schedule of the example with the omission of step 4 (i.e. the delaying of activities with a negative NPV). We observe a capital shortage of 5 at times 5 and 6.

We apply the capital feasibility improvement method and start with the schedule–based variant. We aim to solve the capital conflict at time 5 and can use both activities 2 and 3. set_2 only contains activity 2 but cannot be scheduled later than time 5, so a delay of set_2 cannot solve the capital shortage. set_3 on the contrary contains activities 3, 4, 5 and 6 and can be delayed to time 6 beyond the conflict at time 5. Additional delays until time 8 are, however, required for the set, due to capital shortages. Delaying activity 2 cannot solve any of these conflicts, so no change with respect to activity 2 occurs. No further delays are possible because of the project deadline of 14. Applying the network–based variant of the capital feasibility improvement method can also not solve the capital conflicts. The resulting final infeasible schedule is shown on the bottom left of figure 5.15.

If activity 2 is on the contrary delayed to time 5, set_2 contains both activities 2 and 4 at time 5. Both activities are delayed by 2 time units ($\Delta_{2,next} = 4$, $\Delta_{2,max} = 2$) and the capital shortage at time 5 is solved. Now a shortage at time 7 occurs but this can be solved by delaying set_2 (= {2, 4, 6}) by 2 time units. The resulting and optimal schedule is *C–Feas* and can be found on the bottom right of figure 5.15.

Based on this example, we can conclude that the delays of single activities with a negative NPV are indeed necessary since otherwise we may not be able to solve some capital shortages, as is the case in the bottom left schedule of figure 5.15. These conclusions are furthermore verified in our computational experiments in section **??**.

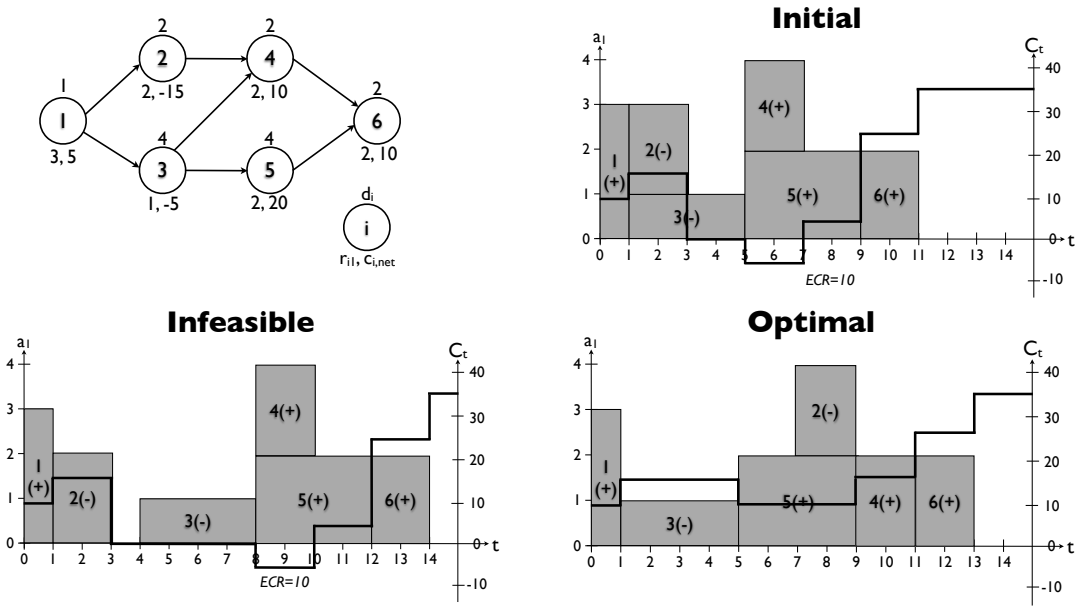


Figure 5.15: Data and schedules example 2.

6

The resource availability cost problem with net present value objective

The resource availability cost problem (RACP) is a variant of the well–known resource–constrained project scheduling problem (RCPSP). Whereas the latter minimizes project makespan given precedence and resource restrictions, the former minimizes total project cost subject to precedence and deadline constraints. We extend the RACP by including cash flows, and optimize the project net present value (NPV). This NPV consists of the activity cash flows discounted to the activity finish times, and the costs of the resource usages. The RACPDC objective allows for a trade–off between the resource usage costs on the one hand and the NPV of individual project activities on the other hand.

In this chapter, we propose to solve the RACP with discounted cash flows (RACPDC) by employing a genetic algorithm and specialized local searches. The added value of the local searches lies in their ability to translate problem characteristics into a good schedule, given the scheduling objective. The local searches are tested in detail for both the RACP and the RACPDC.

6.1 Introduction

In this chapter, the goal is to develop a metaheuristic solution procedure for the resource availability cost problem with discounted cash flows (RACPDC), and with the payments at activities' completion times (PAC) model. The RACPDC is an extension of the resource availability cost problem/resource investment problem (RACP/RIP) first proposed by Möhring (1984). This problem minimizes the total resource usage cost, subject to precedence constraints and a project deadline. This is in contrast with the more extensively studied resource–constrained project scheduling problem (RCPSP), which minimizes the project duration subject to precedence and renewable resource restrictions.

The RACPDC is a practically relevant problem since contractors, or the party responsible for executing a project, only receive a project deadline aside from the project characteristics, i.e. the network structure and required resources per activity. The contractors, however, often have to decide themselves upon the resource capacity assigned to a project. Form this point of view, it makes sense to integrate the decision of the required resource levels and associated costs with a net present value (NPV) objective based on activity cash flows. This way, the overall NPV of the project, including the resource costs, can be evaluated. Additionally, this allows the contractor to consider the trade–off between the employment of additional resource units and maximizing the NPV of the project activities. The resource usage costs are assumed fixed in terms of *timing*, namely at the start of the project, but their *size* can be linked to the project schedule, specifically to the amount of resources required. We extend existing work from chapter 2 in terms of **scheduling technique**, but also analyze the effect of different **solution representations**.

	Timing	**Size**	**Timing & size**
Cash in	Payments at activities' completion times (PAC)	Progress payments (PP)	Progress based payment pattern (PBPP)
		Payments at event occurences (PEO)	Expense based payment pattern (PBPP)
Cash out	Payments at activities' completion times (PAC)	Resource usage costs	General capital model

Figure 6.1: Overview of the research on project scheduling with NPV optimization in chapter 6.

For a recent overview on the RACP literature, we refer to Van Peteghem and Vanhoucke (2013). Since 2013, two more papers have been published on the RACP. The paper of

Qi et al. (2015) proposes a schedule generation scheme for the multi–mode RACP. The authors employ a metaheuristic which is a combination of a particle swarm optimization algorithm and of a scatter search algorithm, and their results are validated on a test set from literature. Shahsavar et al. (2015) use a multi–objective approach to minimize the resource cost, minimize the variability in resource usage, and minimize the project duration. The Pareto–optimal frontier is approximated by making use of three genetic algorithm variations.

To the best of our knowledge no papers exist in literature which discuss the RACPDC. Four papers, however, exist which discuss an extension to the resource renting problem (RRP) (Nübel, 2001). The RRP differs from the RACP based on the resource availability throughout the project. Whereas in the RACP it is assumed that resource levels employed are set for the entire project runtime (Demeulemeester, 1995; Möhring, 1984), in the RRP time–dependent renting costs and time–independent procurement costs are included instead. The recent papers of Najafi and Niaki (2006), Najafi et al. (2009), Najafi and Azimi (2009) and Shahsavar et al. (2010) consider NPV optimization in a RRP context with generalized precedence relations, but incorrectly call the problem the RACP with discounted cash flows. Based on the problem definitions of Möhring (1984), Demeulemeester (1995) and Nübel (2001) we, however, argue that the problem discussed in the papers should be called the resource renting problem with discounted cash flows subject to generalized precedence relations (RRPDC–GPR) (De Reyck and Herroelen, 1998).

The remainder of this chapter is organized as follows. Section 6.2 describes the problem definition of the RACPDC, whereas in section 6.3 we discuss our proposed metaheuristic approach. In section 6.4 we report detailed computational results, compare with existing work, and highlight insights. We finish with a conclusion in section 6.5.

6.2 Problem definition

A directed graph or network $G(N, A)$ can be used to represent a project, with N the nodes or project activities, and A the arcs or precedence relations between the activities. We use the activity–on–the–node (AoN) representation and employ a time–lag of zero for the precedence relations. Each activity i ($i \in N = \{1, \dots n\}$) has a duration d_i, a resource demand r_{ik} for each renewable resource type k ($k \in R = \{1, \dots, |R|\}$), a cash outflow $c_{i,out}$ (< 0) and a cash inflow $c_{i,in}$ (> 0). A start dummy 0 and end dummy activity $n + 1$ are also included, and a project deadline is set equal to δ_{n+1}. The activity finish times f_i are used as decision variables along with the availability a_k of each resource k. Each resource type k furthermore has an associated cost c_k (> 0). Mathematically, the problem can be conceptually formulated as follows:

$$\text{Maximize} \sum_{i=1}^{n} (c_{i,in} + c_{i,out}) \cdot e^{-\alpha f_i} - \sum_{k=1}^{|R|} c_k \cdot a_k \tag{6.1}$$

Subject to:

$$f_i \leq f_j - d_j, \quad \forall (i,j) \in A \tag{6.2}$$

$$\sum_{i \in S(t)} r_{ik} \leq a_k, \quad \forall k \in R, \ t = 1, \ldots, \delta_{n+1} \tag{6.3}$$

$$f_{n+1} \leq \delta_{n+1} \tag{6.4}$$

$$f_i \in int^+, \quad \forall i \in N \tag{6.5}$$

$$a_k \in int^+, \quad \forall k \in R \tag{6.6}$$

In the model, the costs of the resources are assumed to occur at the project start time in the objective function (function (6.1)), hence no discount factor is applied. The cash in– and outflows are both discounted to the activity finish times based on a discount rate α. The precedence relations are imposed in constraints (6.2). The resource restrictions are included in constraints (6.3), with $S(t)$ the set of activities in progress at time t. Constraint (6.4) states the project deadline, and the integrality constraints of the decision variables are modelled in (6.5) and (6.6).

6.3 A genetic algorithm for the RACPDC

In this section, we discuss our proposed genetic algorithm for the RACPDC. We employ a metaheuristic to solve the RACPDC due to three reasons:

1. The RACP (with cost minimization objective instead of NPV maximization) is already difficult to solve. Only the exact procedures of Demeulemeester (1995) and of Rodrigues and Yamashita (2010) exist. The latter improve the results of the former, but state that for projects with 45 activities their procedure could only solve 30% of the test instances. Hence, we believe that a metaheuristic solution procedure is required to solve larger instances of e.g. 60, 90 and 120 activities.

2. The RACPDC is a more complex variant of the RACP due to the non–linear NPV objective. In recent years, metaheuristic approaches have been shown to perform well in solving NPV maximization problems (Vanhoucke, 2010), in particular the genetic algorithms of chapters 2 and 3 (Leyman and Vanhoucke, 2015, 2016b) obtained excellent results.

3. A genetic algorithm has already been successfully applied to several problems from

literature. The decomposition–based genetic algorithm of Debels and Vanhoucke (2007) currently holds the best results for the resource–constrained project scheduling problem (RCPSP), whereas the genetic algorithm of chapter 2 (Leyman and Vanhoucke, 2015) obtained the currently best known solutions for the RCPSP with discounted cash flows (RCPSPDC), and the procedure of chapter 3 (Leyman and Vanhoucke, 2016b) solved the multi–mode RCPSPDC.

We start with preprocessing to determine lower and upper bounds for the required resource levels. We subsequently go into detail about the solution representation and the proposed decoding procedure. Finally, we discuss the different GA operators.

6.3.1 Preprocessing

The preprocessing method of this section determines both a minimum availability a_k^{min} and a maximum availability a_k^{max} for each resource k. The goal of the method is to narrow down the possible feasible values of each resource's availability, without omitting the optimal resource levels.

Minimum resource availability: we employ the same calculations as Van Peteghem and Vanhoucke (2013) for each resource k:

$$a_k^{min} = max\left(\frac{\sum_{i=1}^{n} r_{ik} \cdot d_i}{\delta_{n+1}}, max(r_{ik}|i \in N)\right) \tag{6.7}$$

This minimum availability is subsequently tightened by applying the critical sequence based lower bound of Stinson et al. (1978). The availability is tightened until the lower bound is smaller than or equal to the project deadline.

Maximum resource availability: the cumulative successors and cumulative predecessors are determined for each activity i. We then calculate the total resource usage of type k for each activity i and all activities not included in the set of cumulative successors (CS_i) or the set of cumulative predecessors (CP_i) of activity i. The maximum of these requirements over all activities is used as upper bound for resource type k:

$$a_k^{max} = max\left(\sum_{j=1}^{n}(r_{jk}|j \notin CS_i, j \notin PS_i)|i \in N\right) \tag{6.8}$$

This maximum availability allows for a tighter bound than $\sum_{i=1}^{n} r_{ik}$, without neglecting any precedence feasible solutions. This way, the optimal solution, given all resources, is not omitted.

6.3.2 Solution representation

In terms of solution representation, we distinguish between a topological ordering to determine the order in which the activities are scheduled by the decoding procedure of section 6.3.3, and a resource capacity list which holds the resource availabilities.

Topological ordering (TO): this representation was proposed by Debels et al. (2006) for the RCPSP and holds a priority value for each activity based on the random key (RK) representation. The RK has a size of n+2 (to include both dummies) and the value at position i contains the priority value for activity i. The TO improves the RK representation by avoiding scaling issues, ensuring precedence feasibility, neglecting timing anomalies and solving issues when several activities have the same finish time (Debels et al., 2006).

Resource capacity list (RCL): this list with size $|R|$ holds the resource capacity of each resource r. The values a_k in the list lie within the interval $[a_k^{min}; a_k^{max}]$.

Priority value γ: additionally, we include a binary variable γ, which determines the order in which two parts of the decoding procedure of section 6.3.3 are applied.

6.3.3 Decoding procedure

In this section, we discuss the proposed decoding procedure for the RACPDC. The algorithm is applied for each element (i.e. a TO, RCL and γ) of the metaheuristic's population (section 6.3.4). An overview of the decoding procedure is provided in figure 6.2. In the following subsections we go into detail about the different parts of the method.

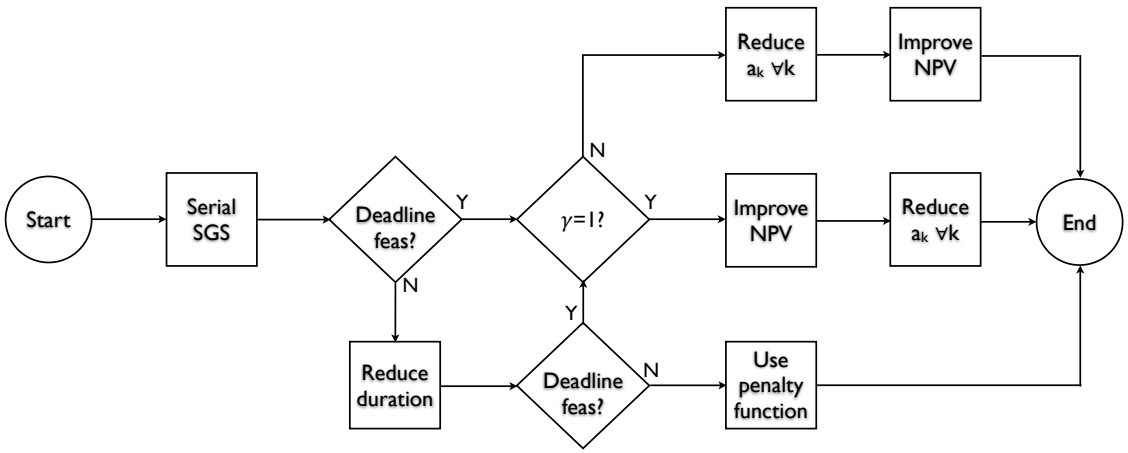


Figure 6.2: Flowchart decoding procedure

6.3.3.1 Initial schedule & deadline feasibility

The first step of the decoding procedure is the construction of a deadline feasible schedule. The serial schedule generation scheme (SGS) (Kelley, 1963) is applied based on the provided TO and RCL (*Serial SGS* in figure 6.2). If the resulting schedule has a project duration larger than the project deadline, the forward–backward improvement method of Li and Willis (1992) is applied to reduce the project duration (*Reduce duration* in figure 6.2). If the schedule is still deadline infeasible, we apply penalty function (6.9) of chapter 2 (Leyman and Vanhoucke, 2015) (*Use penalty function* in figure 6.2) and the decoding procedure terminates.

$$\begin{cases} NPV = NPV_{D-Infeas} \cdot Y_1^{f_{n+1}-\delta_{n+1}} - Y_2 & \text{if } NPV_{D-Infeas} \geq 0 \\ NPV = \frac{NPV_{D-Infeas}}{Y_1^{f_{n+1}-\delta_{n+1}}} - Y_2 & \text{otherwise} \end{cases} \tag{6.9}$$

Penalty function (6.9) reduces the project NPV from function (6.1) ($NPV_{D-Infeas}$) in order to distinguish the infeasible solutions from feasible ones. The project NPV of infeasible solutions is reduced in two ways, to ensure that the NPV of an infeasible solution is considerably worse than that of any feasible solution. Y_1 ($Y_1 \in [0;1]$) reduces $NPV_{D-Infeas}$ based on the difference between the finish time of the end dummy and the project deadline. Y_2 ($Y_2 > 0$) is a large positive value which is subtracted from the project NPV. The variables Y_1 and Y_2 are tested in section 6.4.1.

If a deadline feasible schedule is obtained, either with our without the application of the improvement method of Li and Willis (1992), the decoding procedure continues in one of two ways by evaluating the value of γ. If γ equals 0, the resource usage reduction method of section 6.3.3.2 is first applied and then the NPV improvement method of section 6.3.3.3. If γ is 1, both improvement methods are applied in the reverse order. In either case, both methods are repeated until no further changes occur in activity finish times. Afterwards, the resulting schedule is returned to the metaheuristic.

6.3.3.2 Resource usage reduction

The goal of the resource usage reduction method (*Reduce* a_k $\forall k$ in figure 6.2) is to decrease the total resource usage cost. Since the total cost is, however, determined by the usage of all resources and their corresponding costs, reducing the total cost is no trivial matter. We propose to calculate the total resource cost per time unit TC_t based on the resource usage of type k at time t (u_{kt}): $TC_t = \sum_{k=1}^{|K|} c_k \cdot u_{kt}$. Based on this cost per time unit t a total cost curve can be constructed for the every time unit between 0 and the project deadline. A minimum total cost curve is also constructed: $TC_t^{min} = \sum_{k=1}^{|K|} c_k \cdot a_k^{min}$.

Starting from the activity with the largest TO value, we compute the latest finish time

lf_i based on the finish times of any successors of activity i. We start from this latest finish time and evaluate if this new finish time decreases the objective. If this is the case, we determine the resource feasibility of the delay and remember the delay as best if it is indeed feasible. Either way, we decrease lf_i by 1 and continue with the evaluation of the new lf_i until it equals activity i's current finish time. We then update the schedule based on the best delay found. We continue with the next activity based on the highest TO value of any unconsidered activities, until all activities have been considered. Algorithm 9 provides an overview of the resource usage reduction method.

Algorithm 9 Reduce resource usage

ReduceResUsage ()
  For each activity i starting with the activity with the highest TO value to the lowest
    $u_{kt} = u_{kt} - r_{ik}$, $\forall t \in [f_i - d_i; f_i[$, $\forall k \in R$
    For $t = lf_i$ **to** $f_i + 1$
      $TC_{best} = \sum_{w=f_i-d_i}^{lf_i-1} TC_w$; $f_{i,best} = f_i$
      $u_{kw} = u_{kw} + r_{ik}$, $\forall w \in [t - d_i; t[$, $\forall k \in R$
      If $\sum_{w=f_i-d_i}^{lf_i-1} TC_w < TC_{best}$
        If $u_{kw} \leq a_k$, $\forall w \in [t - d_i; t[$, $\forall k \in R$ **then** $TC_{best} = \sum_{w=f_i-d_i}^{lf_i-1} TC_w$; $f_{i,best} = t$
      End if
    End for
    If $f_{i,best} > f_i$ **then** $f_i = f_{i,best}$
    $u_{kt} = u_{kt} + r_{ik}$, $\forall t \in [f_i - d_i; f_i[$, $\forall k \in R$
  End for
  Return 0

Once algorithm 9 has been completed, we evaluate the maximum resource usage for each resource type k ($max(u_{kt}|t \in \{0, \ldots, \delta_{n+1} - 1\})$). If this maximum is smaller than the corresponding a_k value from the RCL, we update a_k. The objective function value is subsequently updated based on any reductions in RCL and based on the delays of activities.

6.3.3.3 NPV improvement

The goal of the NPV improvement method (*Improve NPV* in figure 6.2) is to delay sets of activities with a negative cumulative NPV. The activity move rules of this section are adapted from the network–based rules of chapter 2 for the RCPSPDC (Leyman and Vanhoucke, 2015). For a deadline feasible schedule these rules aim to improve project NPV, while maintaining feasibility. The move rules recursively consider an activity's immediate predecessors and successors based on the project network. Successors are always added to the set, whereas predecessors are only added if they have a negative net cash flow. Once the set has been constructed, the cumulative NPV of all activities in the set is calculated. If the NPV is negative, a resource feasible delay is applied for the set.

6.3.4 The genetic algorithm

The genetic algorithm (GA) is based on evolutionary biology and was first proposed by Holland (1975). The goal of the algorithm is to improve existing solutions by recombining these solutions into new ones, and uses selection, crossover and mutation operators. We briefly discuss each of the typical GA operators.

Initial population: for the initial population, we randomly generate $2 \cdot |P|$ TO lists, with P the population. In terms of the RCL, we randomly construct a list for each element with values from $[a_k^{min}; a_k^{max}]$ for each resource type k. Finally, a random value γ is also generated for each element. We apply the scheduler of section 6.3.3 to all elements and rank the elements based on their NPV. The best $|P|/2$ elements are added to the population P. The remaining $|P|/2$ positions in P are filled by elements which pass a diversity threshold based on the distance function of Van Peteghem and Vanhoucke (2011):

$$d^f_{p_1,p_2} = \sum_{i=1}^{n} |f_i^{p_2} - f_i^{p_1}| \tag{6.10}$$

In the distance function p_1 and p_2 are two elements which are compared based on the pairwise difference between their corresponding activity finish times. We state that the distance of any element to be added to P in the remaining $|P|/2$ positions, compared to any element already included in P, should be at least $v \cdot n$ ($v \in [0; 1]$). If no $|P|/2$ diverse enough elements could be found in the initial population, we include those elements with the highest diversity.

Selection: this step selects the couples of parents used for crossover. The first parent is the *index*th element of the population P, with *index* a counter for the number of times the selection operator has been called this generation. The second parent is selected based on a four–tournament selection, in which four elements from P are randomly selected. The element with the best objective function value is retained as the second parent. We subsequently label one of both parents randomly as the father and the other as the mother.

Crossover: in this step we create two children based on the parent elements chosen in the selection step. We apply a two–point crossover on each pair of parents, based on random two cut–off points p_1 and p_2. For the first (second) child, all values up to position p_1 and after position p_2 are copied from the father (mother), whereas all values in between both positions are copied from the mother (father). It is furthermore ensured in the second step that no value is included twice, by inserting that the missing elements from the father (mother) in the first (second) child in the order in which they appear in the mother (father). The crossover is applied separately to both the TO and the RCL, with different crossover points. This way, the TO and RCL can be changed independently.

The value for γ is copied from the father for the first child, and from the mother for the second child. The selection and crossover steps are repeated until $|P|$ children have been generated.

Mutation: a mutation operator is included to ensure diversity in the newly generated children by the crossover step. For the TO a two–activity swap is applied with a probability of M_1. In terms of the RCL, we randomly generate a new value from $[a_k^{min}; a_k^{max}]$ for each resource k, with probability M_2.

Population update: this step determines which of the parent elements are retained in the population, and which of the children enter P. On the one hand, the best children should enter P, but on the other hand the best parents should be retained. As such, we choose retain the best R parents and replace the rest of parent population by the best children. Once the population update has been completed, the GA returns to the selection and crossover steps until a stopping criterion has been reached.

6.4 Results

In this section, we discuss our results for the RACPDC. We first configure the proposed algorithm and we compare our results with literature. We finish with some insights. The stopping criterion based on 5,000 generated schedules as defined by Lova et al. (2009) is employed for all tests, and we assume a discount rate of 1%.

We use the RACP30 dataset and the resource costs of Van Peteghem and Vanhoucke (2013). Additionally, we also test our procedure on the PSPLIB's J30, J60, J90 and J120 datasets (Kolisch and Sprecher, 1996) and again use the cost file of Van Peteghem and Vanhoucke (2013). The project deadline is set as follows: $\delta_{n+1} = (1 + \theta) \cdot ef_{n+1}$, with ef_{n+1} the earliest finish time of the project or critical path length, and θ set to 0, 0.10, 0.20, 0.30, 0.40 and 0.50. We generate a net cash flow $c_{i,net}$ for each activity i from the interval [1;500]. Based on a percentage negative (*%Neg*, set to 0, 25, 50, 75 and 100) we randomly assign negative signs to activity cash flows until *%Neg* of the activities have a negative cash flow. This way, the test data include 240 files x 6 deadlines x 5 cash flows = 7,200 instances for RACP30, 480 x 6 x 5 = 14,400 instances for J30, J60 and J90, and 600 x 6 x 5 = 18,000 instances for J120.

In the following subsections, we first configure our algorithm, compare with literature and formulate insights from the best results obtained.

6.4.1 Algorithm configuration

All tests in this section are run on 20% of the RACP30, J30, J60, J90 and J120 datasets, by selecting each first out of five instances.

We wish to determine a suitable value for each of the penalty function and metaheuristic parameters. We test the parameters Y_1 and Y_2 from penalty function (6.9), the population size $|P|$, the parameter v of the initial population, the mutation rates M_1 and M_2 and the number of retained elements R. The Taguchi method (Montgomery, 2005) is used in order to ensure a robust design of experiments. We employ the orthogonal array L'32 for our experiments. The selected values for each parameter are shown in table 6.1, with the best results (highest average NPV, $AvNPV$) marked in bold.

From the table, we can conclude that the greatest effect with respect to penalty function (6.9) is obtained by the parameter Y_2, whereas the parameter Y_1 should simply be large enough to allow for a clear distinction between the NPV of feasible and infeasible solutions. It can also be observed that the mutation rate M_1 for the TO is rather high, whereas M_2 for the RCL is relatively low. The reason for the high value for M_1 lies in the use of a relatively simple mutation operator, i.e. a two–activity swap, and a retention of R elements from the parent population. This way, a higher diversity is required from the offspring in terms of TO value. For the RCL, the employed mutation operator is diverse enough, since it allows for the generation of a new value for each resource k between the lower and upper bounds.

Factor level	Y_1	Y_2	$\|P\|$	v	M_1	M_2	R
1	5,000	0.85	50	0.20	0.85	0.05	1
2	**10,000**	0.90	60	**0.25**	0.90	**0.10**	3
3	15,000	**0.95**	**70**	0.30	**0.95**	0.15	**5**
4	20,000	0.99	80	0.35	0.99	0.20	7

Table 6.1: Parameter testing.

In the remainder of results section, we first analyze our results for the RACP and then for the RACPDC.

6.4.2 Analysis RACP

We compare our algorithm, without the NPV improvement of section 6.3.3.3, with results of the artificial immune system (AIS) of Van Peteghem and Vanhoucke (2013) for the RACP (VP2013). In comparison with our approach, the VP2013 algorithm uses an AIS instead of GA metaheuristic, but also a different scheduler. Their scheduler also uses forward–backward improvement to allow for a shorter makespan within the project

deadline, but uses an extension of the Burgess and Killebrew (1962) algorithm for resource levelling. This extension aims to reduce resource usage instead by means of the objective $\sum_{t=0}^{\delta_n} \sum_{k=1}^{|R|} c_k u_{kt}^2$. Hence, aside from the metaheuristic used, the major difference with our approach lies in the resource usage reduction method.

To allow for a broader comparison, we compare VP2013 and our GA, including the resource usage reduction method of section 6.3.3.2 (TO+RU), with the GA including the adjusted resource levelling method of Van Peteghem and Vanhoucke (2013) instead (TO+BK). Furthermore, we include results for both the finish time list (FTL) and slack list (SL) representations of chapter 4. The latter two are tested without any local search, since the representations allow for a broader range of finish times on their own (see chapter 4), and because this way the scheduler of section 6.3.3 as a whole can be evaluated.

In table 6.2 we compare the five alternatives based on the percentage average improvement from an upper bound (%AvImpr) for the RACP30 dataset. The upper bound is set as the total cost of the resource usage which would allow an implementation of the earliest start schedule (Drexl and Kimms, 2001). The larger the improvement the better the results. The distinction is also made based on the values for the deadline parameter θ. We can conclude that TO+RU outperforms the rest, including VP2013. However, based on an ANOVA analysis we found that the first three alternatives are not statistically significant at the 5% confidence level. As a result, we can state that both the GA and resource usage reduction method show potential (TO+RU better than VP2013 and TO+RU better than TO+BK respectively), but that the added value is limited.

The results for the FTL and SL are significantly worse than those of the other three options, but do not differ statistically with one another. Hence, it is better to use the TO representation with a specialized scheduler instead of the FTL or SL representations of chapter 4.

θ	VP2013	TO+RU	TO+BK	FTL	SL
0	31.19	**31.44**	31.35	28.99	29.14
10	39.05	**39.30**	39.16	34.39	34.49
20	44.74	**44.98**	44.83	38.98	38.84
30	48.53	**48.67**	48.58	42.50	41.31
40	51.36	**51.56**	51.42	45.09	43.56
50	53.55	**53.73**	53.64	47.16	45.62
Overall	44.74	**44.95**	44.83	39.52	38.83

Table 6.2: Comparison RACP30 (%AvImpr).

In table 6.3 the approaches are compared for the PSPLIB datasets based on %AvImpr. We do, however, not posses detailed results of the VP2013 procedure as we do for the RACP30 dataset. Hence, the results are analyzed in an overall manner per PSPLIB

dataset.

We conclude that the difference between both approaches is small, but in favor of the approach in this manuscript for a smaller number of activities, and in favor of Van Peteghem and Vanhoucke (2013) for a larger number of activities. ANOVA analyses, for all but the VP2013 algorithm, showed that the results for TO+RU and TO+BK are not statistically significant at the 5% confidence level, but outperform the FTL and SL approaches.

	VP2013	TO+RU	TO+BK	FTL	SL
J30	?	**44.08**	43.99	37.96	37.23
J60	54.02	**54.18**	54.06	44.11	40.52
J90	**58.78**	58.75	58.57	45.59	41.07
J120	**61.06**	59.20	58.93	44.44	40.11

?: The data reported by Van Peteghem and Vanhoucke (2013) corresponds with our results given a θ=0.

Table 6.3: Comparison PSPLIB (%AvImpr).

To summarize, we can state that the TO+RU performs equally good as the best known procedure in literature of Van Peteghem and Vanhoucke (2013). Given that the goal of this chapter is to propose an algorithm for the RACPDC and not for RACP, we believe that these results are sufficient to show the quality of the proposed method.

6.4.3 Analysis RACPDC

We analyze the effect of the NPV improvement method based on five scenarios, by using the TO+RU approach. We compare results without any NPV improvement (*None*), with the resource usage always first (*RU first*), the NPV improvement always first (*NPV first*), an integrated approach which optimizes the NPV of the activities and resources together (*Combo*), and the complete procedure from figure 6.2 where the factor γ determines the order of both improvement methods (*Gamma* (γ)). We conclude that either *NPV first* or *Gamma* leads to the best results. Furthermore, the results for both alternatives do not differ significantly. This highlights that the focus should be on first optimizing the NPV and only afterwards reducing the resource usage. Alternatively, the results for the *Gamma* option show that it makes sense to let the metaheuristic decide which method to apply first, i.e. the NPV improvement or the resource usage reduction. On average, however, the metaheuristic applies the former in approximately 60% of the schedules first and the latter second. Finally, it can be observed that the *Combo* performs worst of the four improvement options, which shows that it makes more sense to consider optimizing the NPV of the activities and the resource usage costs separately.

	None	*RU first*	*NPV first*	*Combo*	*Gamma (γ)*
RACP30	-130.32	7.32	44.30	-118.10	**47.37**
J30	-79.64	85.18	131.97	-65.47	**133.02**
J60	601.85	1,058.10	**1,154.91**	629.66	1,154.83
J90	1,157.63	1,985.46	**2,126.82**	1,191.67	2,125.82
J120	1,483.89	2,705.90	**2,867.63**	1,509.73	2,866.20

Table 6.4: Local search comparison NPV improvement ($AvNPV$).

Table 6.5 provides an overview of the best results for the RACPDC. As stated earlier (section 6.1), no papers exist in literature which discuss the RACPDC, so we have no point of comparison for the results in the table. We do, however, provide the results in table 6.5 to allow for future comparison, and discuss insights to show more details of our results.

%Neg	RACP30	J30	J60	J90	J120
0	5,366.67	5,162.46	10,443.81	14,960.82	19,230.67
25	2,384.84	2,264.65	5,396.00	8,329.83	11,241.84
50	-74.14	-6.53	906.92	1,676.88	2,062.54
75	-2,440.13	-2,214.52	-2,880.75	-3,760.16	-5,070.14
100	-5,015.72	-4,546.23	-8,091.43	-10,573.28	-13,126.76

Table 6.5: Final results RACPDC ($AvNPV$).

We analyze the effect of the size of the resource usage costs versus the activity cash flows on the importance of both improvement methods. Figure 6.3 displays the average NPV ($AvNPV$, right vertical axis) as the resource usage costs increase. The resource usage costs are increased or decreased with the factors displayed on the horizontal axis. E.g. a factor 10 means that the costs, as defined at the start of section 6.4 are multiplied by 10. Since we employ the methodology with γ as part of the GA, we also plot the curve which displays the average percentage of solutions for which γ is set to zero (first apply the resource usage reduction step, left vertical axis). Based on the average value for γ, we can conclude whether the greatest impact on project NPV comes from the activity cash flows or from the resource usage costs. The results displayed are those for the RACP30 dataset, but similar curves can be constructed for the other datasets.

From the figure, the following can be concluded:

- The effect of increases in the resource costs is larger on the project NPV than on γ, i.e. the order in which both improvement methods should be applied. This can be observed from the figure since the decrease in average project NPV is considerably larger than the increase in %First resource usage. Hence, the activity cash flows have a larger impact on project NPV than the resource costs, even as the latter increase.

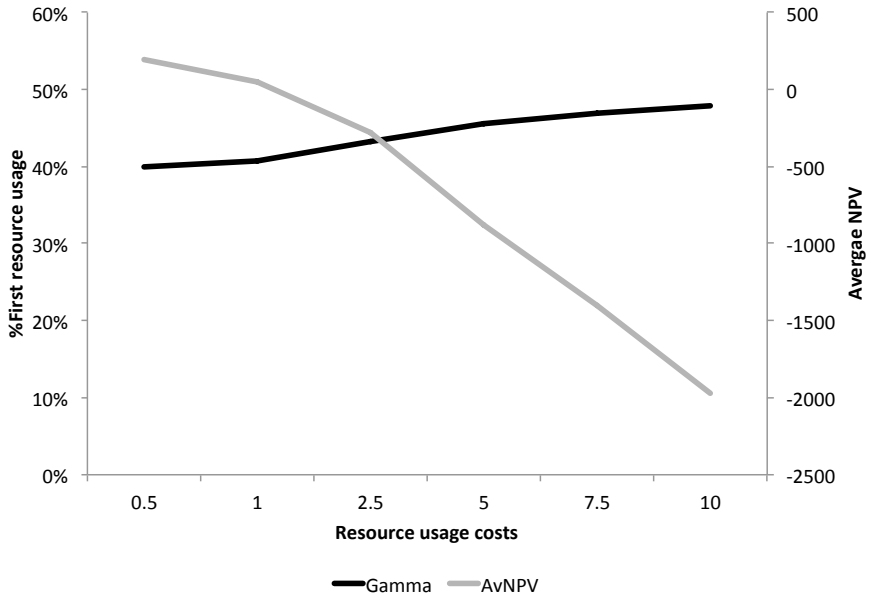


Figure 6.3: Insights

- Even with relatively high resource usage costs (e.g. 10), the NPV improvement method for the activity cash flows remains at least as important as the resource usage reduction (γ=48%).
- With lower resource usage costs (e.g. 5) the focus should be on NPV improvement of the activity cash flows, but reduction of the resource usage costs is still important (γ=40%).

6.5 Conclusions & future research

In this chapter, we have discussed the resource availability cost problem with discounted cash flows (RACPDC). A specialized local search has been proposed as part of a genetic algorithm for the problem, and the approach has been tested on several datasets from literature. Each part of the local search has been validated, and the resource usage reduction part has been compared with literature. The trade–off between the resource usage costs and the activity cash flows has been discussed, along with their impact on the project net present value.

In the future, it may be worthwhile to investigate the impact of the occurrence of the resource usage costs, i.e. when the entire costs are not incurred at the start of the project. Costs could be incurred in a manner similar to the models for cash outflows of chapter 5. A second possible future research avenue concerns solution representation, i.e. the

slack and finish time lists. These representations could be investigated in more detail and integrated with a resource capacity list and a specialized scheduler. Finally, even though it has a clear potential, the proposed resource usage reduction method could be further improved.

7

Conclusions & recommendations for future research

7.1 Conclusions

In this book, we have presented our research on net present value (NPV) optimization in project scheduling. In general, we have discussed multiple models for the timing and/or size of both cash in– and outflows. These models have been heuristically optimized with a strong focus on schedulers and local searches. To allow for an easier understandability of the conclusions discussed here, we have again included the overview figure (figure 7.1).

	Timing	**Size**	**Timing & size**
Cash in	Payments at activities' completion times (PAC)	Progress payments (PP)	Progress based payment pattern (PBPP)
		Payments at event occurences (PEO)	Expense based payment pattern (PBPP)
Cash out	Payments at activities' completion times (PAC)	Resource usage costs	General capital model

Figure 7.1: Overview of the research on project scheduling with NPV optimization.

In **chapter 2**, we have proposed two classes of activity move rules for the payments at activities' completion times (PAC) model, subject to precedence relations, resource restrictions and a project deadline. The goal was to improve the project NPV by moving sets of activities, formed based on either the precedence relations or the neighboring activities in the schedule. The activity move rules have been implemented as part of a metaheuristic, and the results have been shown to outperform two benchmarks from literature.

Chapter 3 extended chapter 2 by including two additional models for cash inflows, namely progress payments (PP) and payments at event occurrences (PEO). These models assume that the cash inflows are received after fixed periods of time instead of at the completion of each activity. The periods of time can furthermore be regular (PP) or irregular (PEO). We have applied these payment models to the problem of chapter 2, and also included its multi–mode variant, which involves the trade–off between different activity modes in terms of duration and resource demands. A more complex scheduling technique than the activity move rules of chapter 2 has been formulated to handle the PP and PEO models, whereas the metaheuristic has been expanded to deal with the problem's multi–mode characteristics. The added value of the proposed techniques is shown based on extensive computational experiments.

Chapter 4 discussed payment models, in which the occurrence time of payments depend on the project schedule (PBPP and EBPP). This is in stark contrast with the

models of chapters 2 and 3, in which only the amount received depended on the schedule. The link between payment times and the project schedule adds an extra dimension to the problem, since changes in activity finish times can have an impact on the timing of payments. Two solution representations have been proposed to handle this increased level of problem complexity, as part of the discrete time/cost trade–off problem. This problem involves trade–offs between different activity modes with different costs and durations for each mode. The proposed solution approach has successfully been compared with a benchmark from literature.

In **chapter 5**, we have extended the model of chapter 2 by imposing capital constraints, which force the cash balance of the project to be positive at each time instance. Three different models, which determine the occurrence of cash outflows, have been discussed as part of a general model for the capital availability. These models determine the times at which cash *outflows* take place, unlike the payment models discussed in chapters 2–4 which focussed on the occurrence of cash *inflows*. In this chapter, we assume that cash inflows are received at activity completion (PAC), as was the case in chapter 2. A new scheduling technique has been proposed to reduce capital shortages by applying two types of delays. The first type aims to delay activities such that their cash outflows are mitigated by cash inflows of other activities, whereas the second type delays activities until succeeding activities are reached to allow for more efficient delays in a later iteration. The scheduler has been applied as part of three metaheuristic approaches from literature, and managerial insights have been provided for contractors.

Finally, **chapter 6** integrates the decision with respect to the required renewable resource level in the NPV objective. This way, the overall NPV of the project, including the resource usage costs, can be evaluated. Additionally, this allows the contractor to consider the trade–off between the employment of additional resource units and maximizing the NPV of the project activities. The resource usage costs are assumed fixed in terms of timing, namely at the start of the project, but their size can be linked to the project schedule, specifically to the amount of resources required. We have successfully compared this approach with the two solution representations of chapter 4, and several insights are given.

With respect to the research questions of chapter 1, we are now able to provide the following answers:

RQ1: What is a good scheduling technique to use for the timing of cash flows?

- *The cumulative NPV of sets of activities should be considered, either based on the project network or based on schedule at hand, to move sets of activities.*

RQ2: What is a good scheduling technique to use for the size of cash inflows?

- *A technique similar as for timing can be applied, but it should be able to take peaks in activity profit curves into account.*
- *If timing and size are considered together, a greater focus should be on the solution representation and metaheuristic, due to the increase in complexity. Otherwise, moves of activity sets may have the opposite effect than what is desired.*

RQ3: How can these schedulers be applied in case of different types of activity trade–offs?

- *The scheduling techniques should be incorporated in an overall framework (metaheuristic), which allows for the selection of modes fitting the problem restrictions.*

RQ4: How can cash outflows and capital be managed for the contractor, under different assumptions?

- *The cash outflows should be incorporated in a general model, to make sure that timing and size can be integrated.*
- *A scheduler should be used, which is able to construct a capital feasible schedule, by allowing for the compensation of cash outflows by cash inflows of other activities.*

RQ5: How can resource usage costs be optimized and integrated in NPV optimization?

- *The resource usage costs should be an integral part of the project NPV, but the trade–off between the resource usage costs and activity cash flows has to be taken into account.*

7.2 Recommendations for future research

Whereas future research recommendations have already been included in some chapters, we here recapitulate the most important ones, along with research avenues which apply to the overall PhD research.

A first major future research avenue concerns the integration of the problems discussed in this book into an overarching framework for NPV optimization in project scheduling. The general model for cash outflows of chapter 5 could be extended to apply to cash inflows as well, which would allow for a more inclusive view on activity cash flows. Multiple activity modes could be used to allow for greater flexibility in scheduling activities, and earliness and tardiness costs could included per activity. This way, the framework should aim to include all possible cash flows concerning a project. One way to provide a scheduler for such a framework, would be to integrate the different scheduling techniques discussed as modules in a general algorithm. This algorithm could then select which modules to use based on the provided problem characteristics, and allow for a decision making tool.

Second, the financing of the contractor could be modelled and analyzed, by considering e.g. loans. In doing so, the general framework could also include not just the optimization of the NPV objective, but also the management of the contractor's entire cash balance. Building on this research avenue, it would be even better to consider financing decisions in a multi–project context, since resources are in general available for the entire company but assigned to specific projects. By taking financing decisions on a multi–project level, a contractor would be able to optimize their entire (renewable, non–renewable and cumulative) resource pool. Such an extension to the framework would ensure a more practically relevant use for the techniques discussed in this PhD.

A third area for future research is the negotiation process with the client concerning the timing and size of payments. Whereas in the research done we assumed that these negotiations had already been completed, it may be worthwhile to analyze this interaction process. Game theory may provide a particular useful avenue, since this would allow different types of client and contractor behaviour. Furthermore, including other restrictions for the contractor, such as the project deadline, would make the negotiation process more realistic. Current work in literature (e.g. Ulusoy and Cebelli, 2000) makes specific assumptions with respect to the behaviour and preferences of both parties, but we believe that this PhD may provide interesting insights and methodologies to improve upon the existing work on negotiation.

References

Aboutalebi, R., Najafi, A., and Ghorashi, B. (2012). Solving multi-mode resource-constrained project scheduling problem using two multi objective evolutionary algorithms. *African Journal of Business Management*, 6(11):4057–4065.

Akkan, C., Drexl, A., and Kimms, A. (2005). Network decomposition–based benchmark results for the discrete time–cost tradeoff problem. *European Journal of Operational Research*, 165:339–358.

Azimi, F., Aboutelabi, R. S., and Najafi, A. A. (2011). Using multi–objective particle swarm optimization for bi–objective multi–mode resource–constrained project scheduling problem. *International Science Index*, 54:242–246.

Baroum, S. and Patterson, J. (1996). The development of cash flow weight procedures for maximizing the net present value of a project. *Journal of Operations Management*, 14:209–227.

Blazewicz, J., Lenstra, J., and Rinnooy Kan, A. (1983). Scheduling subject to resource constraints: notation, classification and complexity. *Discrete Applied Mathematics*, 5:11–24.

Boctor, F. (1993). Heuristics for scheduling projects with resource restrictions and several resource-duration modes. *International Journal of Production Research*, 31:2547–2558.

Brucker, P., Drexl, A., Möhring, W., Neumann, K., and Pesch, E. (1999). Resource-constrained project scheduling: Notation, classification, models, and methods. *European Journal of Operational Research*, 112:3–41.

Burgess, A. and Killebrew, J. (1962). Variation in activity level on a cyclic arrow diagram. *Industrial Engineering*, 2:76–83.

Chen, A. and Chyu, C. (2008). A memetic algorithm for maximizing net present value in resource-constrained project scheduling problem. *IEEE Congres on Evolutionary Computation*, pages 2401–2408.

Chen, W.-N. and Zhang, J. (2012). Scheduling multi-mode projects under uncertainty to optimize cash flows: A monte carlo ant colony system approach. *Journal of Computer Science and Technology*, 27(5):950–965.

Chen, W.-N., Zhang, J., Chung, H., Huang, R.-Z., and Liu, O. (2010). Optimizing discounted cash flows in project scheduling - an ant colony optimization approach. *IEEE Transactions on Systems, Man., and Cybernetics - Part C: Applications and Reviews*, 40:64–77.

Damak, N., Jarboui, B., Siarry, P., and Loukil, T. (2009). Differential evolution for solving multi-mode resource-constrained project scheduling problems. *Computers and Operations Research*, 36:2653–2659.

Dayanand, N. and Padman, R. (1997). On modelling payments in project scheduling. *Management Science*, 48(9):906–918.

Dayanand, N. and Padman, R. (2001a). Project contracts and payment schedules: the client's problem. *Management Science*, 47(12):1654–1667.

Dayanand, N. and Padman, R. (2001b). A two stage heuristic for scheduling payments in projects. *Annals of Operations Research*, 102:197–220.

De Reyck, B. and Herroelen, W. (1998). An optimal procedure for the resource-constrained project scheduling problem with discounted cash flows and generalized precedence relations. *Computers and Operations Research*, 25:1–17.

Debels, D., De Reyck, B., Leus, R., and Vanhoucke, M. (2006). A hybrid scatter search/-electromagnetism meta–heuristic for project scheduling. *European Journal of Operational Research*, 169:638–653.

Debels, D. and Vanhoucke, M. (2007). A decomposition–based genetic algorithm for the resource–constrained project scheduling problem. *Operations Research*, 55(3):457–469.

Demeulemeester, E. (1995). Minimizing resource availability costs in time–limited project networks. *Management Science*, 41(10):1590–1598.

Demeulemeester, E., De Reyck, B., Foubert, B., Herroelen, W., and Vanhoucke, M. (1998). New computational results for discrete time/cost trade–off problem in project networks. *Journal of the Operational Research Society*, 49:1153–1163.

Demeulemeester, E. and Herroelen, W. (1992). A branch-and-bound procedure for the multiple resource-constrained project scheduling problem. *Management Science*, 38:1803–1818.

Demeulemeester, E., Vanhoucke, M., and Herroelen, W. (2003). Rangen: A random network generator for activity-on-the-node networks. *Journal of Scheduling*, 6:17–38.

Doersch, R. and Patterson, J. (1977). Scheduling a project to maximize its net present value: A zero-one programming approach. *Management Science*, 23(8):882–889.

Drexl, A. and Kimms, A. (2001). Optimization guided lower and upper bounds for the resource investment problem. *Journal of the Operational Research Society*, 52(3):340–351.

Erengüç, S., Tufekci, S., and Zappe, C. (1993). Solving time/cost trade–off problems with discounted cash flows using generalized benders decomposition. *Naval Research Logistics*, 40:25–50.

Etgar, R. and Shtub, A. (1999). Scheduling project activities to maximize the net present value - the case of linear time-dependant cash flows. *International Journal of Production Research*, 37(2):329–339.

Gendreau, M. and Potvin, J.-Y. (2010). Tabu search. In Gendreau, M. and Potvin, J.-Y., editors, *Handbook of metaheuristics*, pages 41–59. Springer.

Gu, H., Schutt, A., and Stuckey, P. (2013). A Lagrangian relaxation based forward-backward improvement heuristic for maximising the net present value of resource-constrained projects. *Lecture Notes in Computer Science*, 7874:340–346.

Gu, H., Stuckey, P., and Wallace, M. (2012). Maximising the net present value of large resource-constrained projects. *Lecture Notes in Computer Science*, 7514:767–781.

Hartmann, S. and Briskorn, D. (2010). A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*, 207:1–14.

Hazir, O., Haouari, M., and Erel, E. (2010). Discrete/time cost trade–off problem: A decomposition–based solution algorithm for the budget version. *Computers & Operations Research*, 37:649–655.

He, Z., Liu, R., and Jia, T. (2012). Metaheuristics for multi-mode capital-constrained project payment scheduling. *European Journal of Operational Research*, 223:605–613.

He, Z., Liu, R., and Xu, Y. (2009a). Client perspective based multimode project payment scheduling problem and its heuristic algorithm. *Systems Engineering – Theory & Practice*, 29(2):70–77.

He, Z., Wang, N., Jia, T., and Xu, Y. (2009b). Simulated annealing and tabu search for multi–mode project payment scheduling. *European Journal of Operational Research*, 198:688–696.

He, Z., Wang, N., and Li, P. (2014). Simulated annealing for financing cost distribution based project payment scheduling from a joint perspective. *Annals of Operations Research*, 213:203–220.

He, Z. and Xu, Y. (2008). Multi–mode project payment scheduling problem with bonus–penalty structure. *European Journal of Operational Research*, 189:1191–1207.

Herroelen, W., De Reyck, B., and Demeulemeester, E. (1998). Resource-constrained project scheduling: A survey of recent developments. *Computers and Operations Research*, 25:279–302.

Herroelen, W., Demeulemeester, E., and De Reyck, B. (1999). A classification scheme for project scheduling. In Weglarz, J., editor, *Project Scheduling - recent models, algorithms and applications. International Series in Operations Research and Management Science*, number 14, pages 77–106. Boston: Kluwer Academic Publishers.

Herroelen, W., Van Dommelen, P., and Demeulemeester, E. (1997). Project networks with discounted cash flows: A guided tour through recent developments. *European Journal of Operational Research*, 100:97–121.

Holland, J. (1975). *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor.

Hosseini, Z. S., Pour, J. H., and Roghanian, E. (2014). A bi–objective pre–emption multi-mode resource constrained project scheduling problem with due dates in the activities. *Journal of Optimization in Industrial Engineering*, 15:15–25.

Icmeli, O. and Erengüç, S. (1996). A branch and bound procedure for the resource-constrained project scheduling problem with discounted cash flows. *Management Science*, 42(10):1395–1408.

Józefowska, J., Mika, M., Rózycki, R., Waligóra, G., and Weglarz, J. (2001). Simulated annealing for multi-mode resource-constrained project scheduling. *Annals of Operations Research*, 102:137–155.

Kavlak, N., Ulusoy, G., Sivrikaya-Şerifoğlu, F., and Birbil, I. (2009). Client-contractor bargaining on net present value in project scheduling with limited resources. *Naval Research Logistics*, 56:93–112.

Kazaz, B. and Sepil, C. (1996). Project scheduling with discounted cash flows and progress payments. *Journal of the Operational Research Society*, 42:1262–1272.

Kazemi, F. and Tavakkoli-Moghaddam, R. (2010). Solving a multi–objective multi–mode resource–constrained project scheduling problem with discounted cash flows. *6th International Project Management Conference*.

Kelley, J. J. (1963). The critical-path method: Resources planning and scheduling. In Muth, J. and Thompson, J., editors, *Industrial Scheduling*, pages 347–365. Prentice-Hall, New Jersey.

Kimms, A. (2001). Maximizing the net present value of a project under resource constraints using a Lagrangian relaxation based heuristic with tight upper bounds. *Annals of Operations Research*, 102:221–236.

Kolisch, R. (1996). Serial and parallel resource-constrained project scheduling methods revisited: Rules and computation. *European Journal of Operational Research*, 90:320–333.

Kolisch, R. and Drexl, A. (1997). Local search for nonpreemptive multi-mode resource-constrained project scheduling. *IIE Transactions*, 29:987–999.

Kolisch, R. and Hartmann, S. (1999). Heuristic algorithms for solving the resource- constrained project scheduling problem: Classification and computational analysis. *European Journal of Operational Research*, 112:3–41.

Kolisch, R. and Sprecher, A. (1996). PSPLIB - A project scheduling problem library. *European Journal of Operational Research*, 96:205–216.

Leyman, P., Van Driessche, N., and Vanhoucke, M. (2016). Metaheuristics for the discrete time/cost trade–off problem with net present value optimization and different payment models. *Working paper*.

Leyman, P. and Vanhoucke, M. (2015). A new scheduling technique for the resource–constrained project scheduling problem with discounted cash flows. *International Journal of Production Research*, 53(9):2771–2786.

Leyman, P. and Vanhoucke, M. (2016a). Capital– and resource–constrained project scheduling with net present value optimization. *European Journal of Operational Research*, Article in press.

Leyman, P. and Vanhoucke, M. (2016b). Payment models and net present value optimization for resource-constrained project scheduling. *Computers & Industrial Engineering*, 91:139–153.

Leyman, P. and Vanhoucke, M. (2016c). The resource availability cost problem with net present value objective. *Working paper*.

Li, K. and Willis, R. (1992). An interative scheduling technique for resource-constrained project scheduling. *European Journal of Operational Research*, 56:370–379.

Lova, A., Tormos, P., Cervantes, M., and Barber, F. (2009). An efficient hybrid genetic algorithm for scheduling projects with resource constraints and multiple execution modes. *International Journal of Production Economics*, 117:302–316.

Martí, R., Laguna, M., and Glover, F. (2006). Principles of scatter search. *European Journal of Operational Research*, 169:359–372.

Mika, M., Waligóra, G., and Weglarz, J. (2005). Simulated annealing and tabu search for multi-mode resource-constrained project scheduling with positive discounted cash flows and different payment models. *European Journal of Operational Research*, 164:639–668.

Möhring, R. (1984). Minimizing costs of resource requirements in project networks subject to a fixed completion time. *Operations Research*, 31(1):89–120.

Möhring, R., Shulz, A., Stork, F., and Uetz, M. (2001). On project scheduling with irregular starting time costs. *Operations Research Letters*, 28:149–154.

Möhring, R., Shulz, A., Stork, F., and Uetz, M. (2003). Solving project scheduling problems by minimum cut computations. *Management Science*, 49(3):330–350.

Montgomery, D. (2005). *Design and Analysis of Experiments*. John Wiley and Sons Inc., Hoboken, New Jersey.

Najafi, A. and Azimi, F. (2009). A priority rule–based heuristic for resource investment project scheduling problem with discounted cash flows and tardiness penalties. *Mathematical Problems in Engineering*, 2009:1–10.

Najafi, A. and Niaki, S. (2006). A genetic algorithm for the resource investment problem with discounted cash flows. *Applied Mathematics and Computation*, 183:1057–1070.

Najafi, A., Niaki, S., and Shahsavar, M. (2009). A parameter–tuned genetic algorithm for the resource investment problem with discounted cash flows and generalized precedence relations. *Computers & Operations Research*, 36:2994–3001.

Neumann, K. and Schwindt, C. (2002). Project scheduling with inventory constraints. *Mathematical Methods of Operations Research*, 56:513–533.

Nübel, H. (2001). The resource renting problem subject to temporal constraints. *OR Spectrum*, 23:359–381.

Özdamar, L. (1998). On scheduling project activities with variable expenditure rates. *IIE Transactions*, 30:695–704.

Özdamar, L. and Dündar, H. (1997). A flexible heuristic for a multi–mode capital constrained project scheduling problem with probabilistic cash inflows. *Computers & Operations Research*, 24(12):1187–1200.

Patterson, J. (1976). Project scheduling: The effects of problem structure on heuristic scheduling. *Naval Research Logistics*, 23:95–123.

PMBOK (2004). *A Guide to the Project Management Body of Knowledge, Third Edition*. Newton Square, Pa.: Project Management Institute, Inc.

Pritsker, A., Watters, L., and Wolfe, P. (1969). Multiproject scheduling with limited resources: A zero-one programming approach. *Management Science*, 16(1):93–108.

Qi, J.-J., Liu, Y.-J., Jiang, P., and Guo, B. (2015). Schedule generation scheme for solving multi–mode resource availability cost problem by modified particle swarm optimization. *Journal of Scheduling*, 18:285–298.

Reeves, C. (2010). Genetic algorithms. In Gendreau, M. and Potvin, J.-Y., editors, *Handbook of metaheuristics*, pages 109–139. Springer.

Resende, M., Ribeiro, C., Glover, F., and Martí, R. (2010). Scatter search and path–relinking: Fundamentels, advances, and applications. In Gendreau, M. and Potvin, J.-Y., editors, *Handbook of metaheuristics*, pages 87–107. Springer.

Rodrigues, S. and Yamashita, D. (2010). An exact algorithm for minimizing resource availability costs in project scheduling. *European Journal of Operational Research*, 206:562–568.

Russell, A. (1970). Cash flows in networks. *Management Science*, 16:357–373.

Russell, R. (1986). A comparison of heuristics for scheduling projects with cash flows and resource restrictions. *Management Science*, 32:1291–1300.

Schutt, A., Chu, G., Stuckey, P., and Wallace, M. (2012). Maximising the net present value for resource-constrained project scheduling. *Lecture Notes in Computer Science*, 7298:362–378.

Schwindt, C. and Zimmermann, J. (2001). A steepest ascent approach to maximizing the net present value of projects. *Mathematical Methods of Operations Research*, 53:435–450.

Seifi, M. and Tavakkoli-Moghaddam, R. (2008). A new bi-objective model for a multi-mode resource-constrained project scheduling problem with discounted cash flows and four payment models. *IJE Transactions A: Basics*, 21(4):347–360.

Selle, T. and Zimmermann, J. (2003). A bidirectional heuristic for maximizing the net present value of large-scale projects subject to limited resources. *Naval Research Logistics*, 50:130–148.

Sepil, C. and Ortac, N. (1997). Performance of the heuristic procedures for constrained projects with progress payments. *The Journal of the Operational Research Society*, 48(11):1123–1130.

Shahsavar, A., Najafi, A., and Niaki, S. (2015). Three self–adaptive multi–objective evolutionary algorithms for a triple–objective project scheduling problem. *Computers & Industrial Engineering*, 87:4–15.

Shahsavar, M., Niaki, S., and Najafi, A. (2010). An efficient genetic algorithm to maximize net present value of project payments under inflation and bonus–penalty policy in resource investment problem. *Advances in Engineering Software*, 41:1023–1030.

Shtub, A. and Etgar, R. (1997). A branch and bound algorithm for scheduling projects to maximize net present value: the case of time dependent, contingent cash flows. *International Journal of Production Research*, 35(12):3367–3378.

Smith-Daniels, D. and Aquilano, N. (1987). Using a late-start resource-constrained project schedule to improve project net present value. *Decision Sciences*, 18:617–630.

Smith-Daniels, D., Padman, R., and Smith-Daniels, V. (1996). Heuristic scheduling of capital constrained projects. *Journal of Operations Management*, 14:241–254.

Smith-Daniels, D. and Smith-Daniels, V. (1987). Maximizing the net present value of a project subject to materials and capital constraints. *Journal of Operations Management*, 7(1-2):33–45.

Sprecher, A., Hartmann, S., and Drexl, A. (1997). An exact algorithm for project scheduling with multiple modes. *OR Spektrum*, 19:195–203.

Stinson, J., Davis, E., and Khumawala, B. (1978). Multiple resource-constrained scheduling using branch-and-bound. *IIE Transactions*, 10:252–259.

Sung, C. and Lim, S. (1994). A project activity scheduling problem with net present value measure. *International Journal of Production Economics*, 37:177–187.

Tavares, L., Antunes Ferreira, J., and Silva Coelho, J. (1998). On the optimal management of project risk. *European Journal of Operational Research*, 107:451–469.

Ulusoy, G. and Cebelli, S. (2000). An equitable approach to the payment scheduling problem in project management. *European Journal of Operational Research*, 127:262–278.

Ulusoy, G., Sivrikaya-Şerifoğlu, F., and Şahin, S. (2001). Four payment models for the multi-mode resource constrained project scheduling problem with discounted cash flows. *Annals of Operations Research*, 102:237–261.

Valls, V., Ballestín, F., and Quintanilla, S. (2004). A population–based approach to the resource–constrained project scheduling problem. *Annals of Operations Research*, 131:305–324.

Valls, V., Quintanilla, S., and Ballestín, F. (2003). Resource–constrained project scheduling: A critical activity reordering heuristic. *European Journal of Operational Research*, 149:282–301.

Van Peteghem, V. and Vanhoucke, M. (2010). A genetic algorithm for the preemptive and non-preemptive multi-mode resource-constrained project scheduling problem. *European Journal of Operational Research*, 201:409–418.

Van Peteghem, V. and Vanhoucke, M. (2011). Using resource scarceness characteristics to solve the multi-mode resource-constrained project scheduling problem. *Journal of Heuristics*, 17:705–728.

Van Peteghem, V. and Vanhoucke, M. (2013). An artificial immune system algorithm for the resource availability cost problem. *Flexible Services and Manufacturing Journal*, 25:122–144.

Van Peteghem, V. and Vanhoucke, M. (2014). An experimental investigation of metaheuristics for the multi-mode resource-constrained project scheduling problem on new dataset instances. *European Journal of Operational Research*, 235:62–72.

Vanhoucke, M. (2006). An efficient hybrid search procedure for various optimization problems. *Lecture Notes in Computer Science*, 5482:13–24.

Vanhoucke, M. (2009). A genetic algorithm for net present value maximization for resource constrained projects. *Lecture Notes in Computer Science*, 5482:13–24.

Vanhoucke, M. (2010). A scatter search procedure for maximizing the net present value of a resource-constrained project with fixed activity cash flow. *International Journal of Production Research*, 48(7):1983–2001.

Vanhoucke, M. (2012). *Project Management with Dynamic Scheduling – Baseline Scehduling, Risk Analysis and Project Control*. Springer–Verlag Berlin Heidelberg.

Vanhoucke, M. and Debels, D. (2007). The discrete time/cost trade–off problem: extensions and heuristic procedures. *Journal of Scheduling*, 10:311–326.

Vanhoucke, M., Demeulemeester, E., and Herroelen, W. (2001a). Maximizing the net present value of a project with linear time-dependant cash flows. *International Journal of Production Research*, 39:3159–3181.

Vanhoucke, M., Demeulemeester, E., and Herroelen, W. (2001b). On maximizing the net present value of a project under renewable resource constraints. *Management Science*, 47(8):1113–1121.

Vanhoucke, M., Demeulemeester, E., and Herroelen, W. (2003). Progress payments in project scheduling problems. *European Journal of Operational Research*, 148:604–620.

Yang, K., Tay, L., and Sum, C. (1995). A comparison of stochastic scheduling rules for maximizing project net present value. *European Journal of Operational Research*, 85:327–339.

Zhu, D. and Padman, R. (1999). A metaheuristic scheduling procedure for resource-constrained projects with cash flows. *Naval Research Logistics*, 46:912,927.